TECHNICAL MANUAL

OPERATION AND MAINTENANCE INSTRUCTIONS WITH SOFTWARE DOCUMENTATION

FOR

RATE OF FLOW TEST STAND
PART NUMBER 10047-10
NSN NUMBER 4920-01-201-1593

S/N: 002

APS SYSTEMS
CONTRACT NUMBER F41608-88-C-2452

Distribution Statement - This publication is required for official use or for administrative or operational purposes only. Distribution is limited to U.S. Government agencies. Other requests for this document must be referred to San Antonio ALC/MMEDT, Kelly AFB, TX, 78241-5990.

15MAY 1991

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

Tables **Page**

GENERAL MACHINE SAFETY GUIDELINES

Operators and service personnel shall read the entire manual for this test stand before beginning to operate or service the stand. Familiarization with the manual will minimize exposure to risk of accidents or injury. The procedures given in this manual have proved safe; APS Systems, assumes no responsibility for personal injury or damage to equipment resulting from the application of these procedures. All personnel shall read this manual; upon installation at site, personnel shall review all safety instructions provided by the company. In this manual, it is not possible to address all safety considerations. Therefore, always be alert and work safely. Only qualified operators and service personnel are authorized to perform the procedures given in this manual. Know and obey the following warnings and safety considerations at all times:

a. Know the location of the EMERGENCY STOP pushbuttons before operating or servicing the test stand.
b. At all times, wear safety glasses approved by your company.
c. Secure loose clothing, neckties and long hair. Remove items such as rings, necklaces, bracelets and wrist watches to avoid their entanglement with moving parts. Wear medical alert jewelry with caution.
d. Never operate solenoid valves, limit switches or proximity switches by hand. This practice can cause dangerous, unexpected mechanical movements.
e. Close all doors on the electrical enclosure during normal operation.
f. Keep the operator's work area clean. Do not work on slippery floors and surfaces. Remove all chips and hazardous obstacles.
g. Before performing any service or interior maintenance, stop the test stand completely by disengaging the electrical disconnect switch.
h. Observe power lockout procedures according to federal, state, or company standards.
i. Do not smoke or cause sparks or open flame around the test stand at any time, particularly during a test.
j. Read and observe all signs posted on the test stand.

SAFETY GUIDELINES FOR TROUBLESHOOTING, REPAIR AND MAINTENANCE.

All troubleshooting and repair shall be performed by qualified and properly-trained personnel. The following safety considerations shall be observed at all times:

a. Before troubleshooting or servicing the test stand, have an up-to-date set of electrical, hydraulic, and pneumatic drawings.

b. Remove rings, necklaces, watches and jewelry, and all such metallic items. They increase electrical hazards. Medical alert jewelry shall be worn with caution.

c. Wear safety glasses which feature non-conducting components. Avoid glasses which have metal rims or metal side shields.

WARNING

Troubleshooting with the power on is dangerous. Be extremely careful.

d. It may be necessary to troubleshoot equipment with the power on. In these instances, only those panels, doors, or covers need to be opened. Know what voltage is present at all points before troubleshooting.

e. When repair is completed, make sure all fittings and connections have been tightened before turning on the supply of compressed air.

RESERVOIR SIDE SECTION TO SINK SIDE SECTION CONNECTIONS

The Test Stand is being shipped in two sections - Reservoir side section and Sink side section (already attached to the electrical control cabinet). Connect the reservoir side section to the sink side section with splice plates and secure with bolts. Two splice plates along with eight bolts are shipped in a carton. Remove splice plates and place on the sides at the bottom and secure with bolts in pre-tapped holes in the two side sections. All pipes, fittings, and electrical wires have been tagged/numbered on both sections. Connect the same numbers at both sections.

AIR CONNECTIONS

The air connections are numbered numerically - 1 through 3. Connect B-nuts on Reservoir side (Figure 1) and Sink side (Figure 2) to their respective fittings.

FLUID CONNECTIONS

The fluid connections are numbered by letters A, B, and C.

1. Insert gasket (shipped in a carton) between flanges at A and secure with bolts, washers, and nuts.

2. Connect B-nut for B on Sink side (Figure 2) to fitting at Reservoir side (Figure 1).

3. Remove two gaskets and check valve 505 from the carton. Attach one gasket and check valve at C on Sink side (Figure 2) flange. Connect other side of check valve and the other gasket to the flange C at Reservoir side (Figure 1) and secure with bolts, washers, and nuts.

ELECTRICAL CONNECTIONS

The electrical conduits are numbered E1, E2, and E3. Wires are numbered and tagged also.

1. Wires from E1 have been pulled out at Sink side section (Figure 2). Connect these in accordance with Figure 3 and electrical schematic Figure 4-2.

2. Wires from E2 have been pulled out at Sink side section (Figure 2). Connect these in accordance with Figure 4 and electrical schematic Figure 4-2.

3. Wires from E3 have been pulled out at Sink side section (Figure 2). Connect these in accordance with Figure 5 and electrical schematic Figure 4-2.

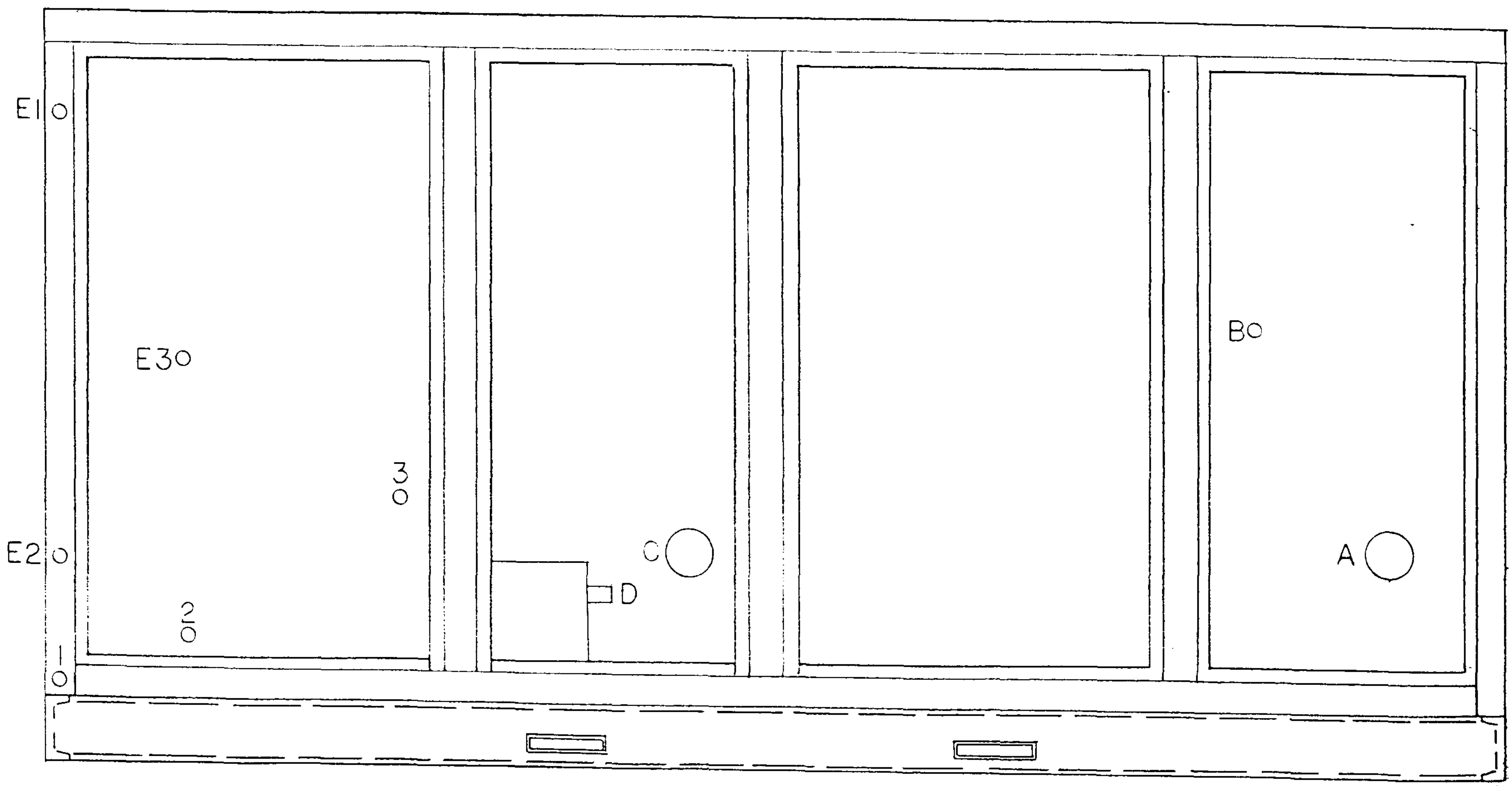


FIGURE 1. RESERVOIR SIDE SECTION

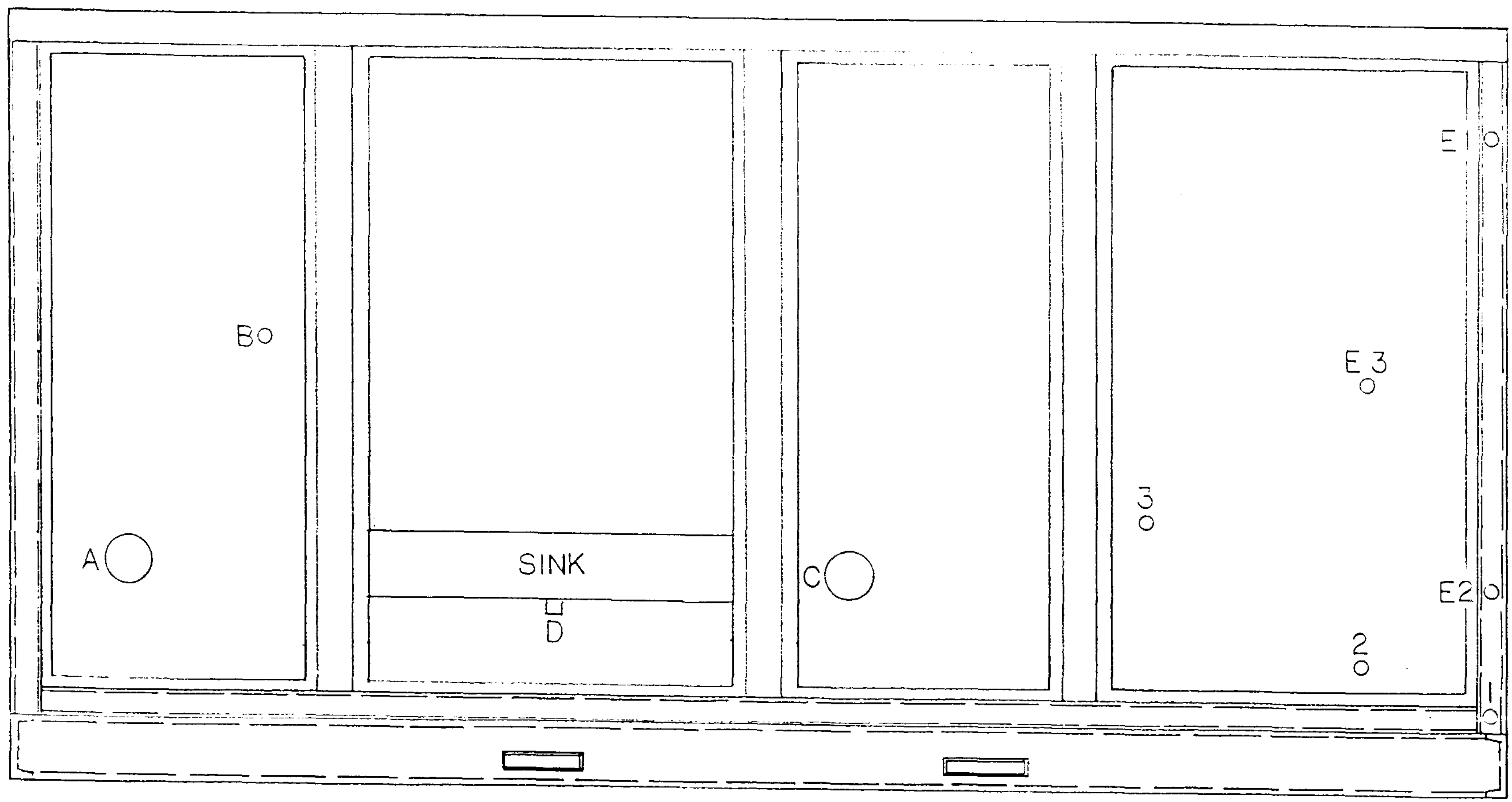


FIGURE 2. SINK SIDE SECTION

C

FIGURE 3

SINK SIDE SECTION TO RESERVOIR SIDE SECTION WIRING INTERCONNECTION LIST

SINK SECTION WIRE NUMBER	TO DESTINATION	OTHER END TERMINATION (FOR REFERENCE ONLY)
1A	Door Interlock E548-2	E553, 10A Fuse
1B	Door Interlock E548-2	E549
1B	Internal Light E549	E548-2
2	Internal Light E549	TB550-1 Any Lug
2	Air Solenoid E504	TB550-1 Any Lug
10	Air Solenoid E504	TB552-10

FIGURE 4

SINK SIDE SECTION TO RESERVOIR SIDE SECTION WIRING INTERCONNECTION LIST

SINK SECTION WIRE NUMBER	TO DESTINATION	OTHER END TERMINATION (FOR REFERENCE ONLY)
13	Residue Level E507	TB550-2 Any Lug
14A	Residue Level E507	TB550-3-12
24	Facility Air Pressure E513	TB550-4-1
25	Facility Air Pressure E513	TB550-4-2
25	Fuel Temperature E514	TB550-4-2
26	Fuel Temperature E514	TB550-4-3
26	Reservoir Level Switch E515	TB550-4-3
27	Reservoir Level Switch E515	TB550-4-4
13	Pump Contactor E516	TB550-2 Any Lug
20	Pump Contactor E516	TB550-3-17
21	Pump Overload E517	TB550-4-11
29	Pump Overload E517	TB550-3-18

FIGURE 5

SINK SIDE SECTION TO RESERVOIR SIDE SECTION
WIRING INTERCONNECTION LIST

SINK SECTION WIRE NUMBER	TO DESTINATION	OTHER END TERMINATION (FOR REFERENCE ONLY)
13	Water Pressure E512	TB550-2 Any Lug
24	Water Pressure E512	TB550-4-1

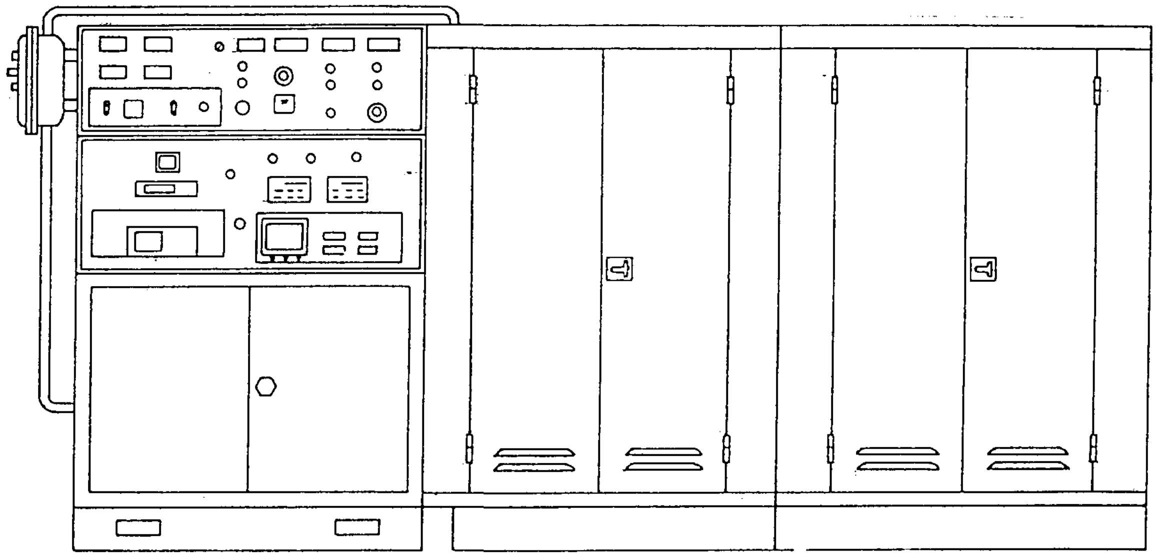


Figure 1-1. Fuel Rate of Flow Test Stand (Sheet 1 of 2)

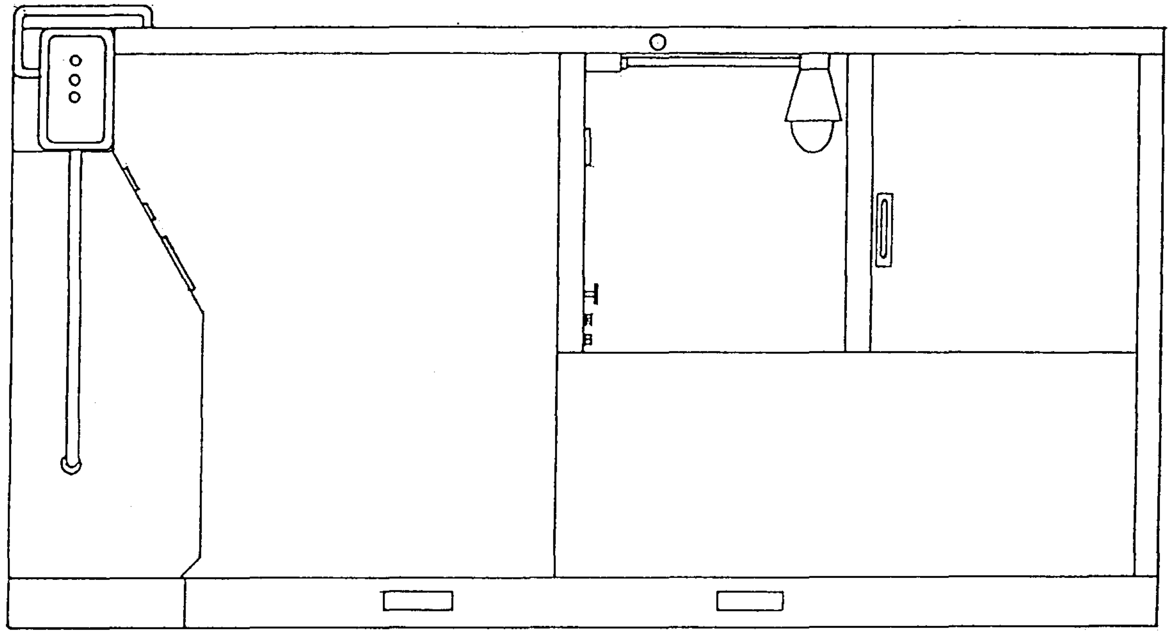


Figure 1-1. Fuel Rate of Flow Test Stand (Sheet 2 of 2)

CHAPTER I

INTRODUCTION AND DESCRIPTION

1-1 INTRODUCTION

This manual contains descriptive information, operation, maintenance and troubleshooting instructions, drawings and spare parts lists for the Fuel Rate of Flow Test Stand Part No. 10047-10, serial numbers 001 and 002, manufactured by APS Systems, 3535 West Fifth Street, Oxnard, California 93030-6498, U.S.A. Relative position locations (front, rear, left and right) are from the operator's position. Components referred to by an item number will appear by that number throughout this manual. These numbers correlate with the numbers on the test stand name plates, tags and the drawings.

1-2 PURPOSE

The Fuel Rate of Flow Test Stand (Figure 1-1) is a precision system for testing and calibrating fuel flow transmitters for various aircraft engines against pre-calibrated standard flowmeters. The test stand is capable of testing the following flow transmitters (Table 1-1):

Table 1-1. Flow Transmitters

PART NO.	COVERED IN TECH. ORDER	INTERFACE DEFINITION	TYPE	FLOW RANGE IN PPH
8TJ50GAS4	5L13-2-12-3	ID-T-106	MB-2	200-12,000
8TJ50GAS5	5L13-2-12-3	ID-T-106	MB-2	200-12,000
8TJ50GBM5	5L13-2-12-3	ID-T-104	TR6-7/A	300-15,000
8TJ62GBA3	5L13-2-16-3	ID-T-107	TRU-27/A	500-20,000
8TJ62GBC3	5L13-2-19-3	ID-T-115	TRU-63A	500-80,000
8TJ62GBK3	5L13-2-16-3	ID-T-105	TRU-27/A	500-20,000
8TJ62GBZ3	5L13-2-19-3	ID-T-113	TRU-63/A3	500-80,000
8TJ62GCA3	5L13-2-21-13	ID-T-111	TRU-63/A-4	500-100,000
8TJ64GDM2	5L13-3-19-3	ID-T-102	TRU-31A/A-8	200-2,500
8TJ85GHM2	5L13-2-20-3	ID-T-101	TRU-98/A-7	200-15,000
9115-16A1A	5L13-2-8-33	ID-T-109	J-6A	200-12,000
9115-16B1A	5L13-2-8-23	ID-T-110	J-6A	600-12,000
9115-16C1A	5L13-2-8-23	ID-T-110	J-6A	600-12,000
9115-16C4A	5L13-2-8-23	ID-T-110	J-6A	600-12,000
9115-16D1	5L13-2-8-23	ID-T-110	J-6A	600-12,000
9117-16A1	5L13-2-10-3	ID-T-103	J7	200-12,000
9121-21A1	5L13-2-8-13	ID-T-114	MA1	400-24,000
150-005-003	5L13-2-21-3	ID-T-112	TRU-63/A-4	500-100,000
9-196-01	5L13-2-23-2	ID-T-108	N/A	650-14,000

1-3 SAFETY PRECAUTIONS

Warnings and cautions appearing throughout this technical manual are of paramount importance to personnel and equipment safety. Prior to any attempt to operate, maintain, troubleshoot, or repair any part of the Fuel Rate of Flow test stand system, all warnings and cautions should be thoroughly reviewed and understood. Refer to the Safety Summary which appears in the front matter of this manual. The following paragraphs define warnings, cautions, and notes as they are used in the manual.

WARNING

Identifies an operating or maintenance procedure, practice, condition, statement, etc, which if not strictly followed could result in death or serious injury to personnel.

CAUTION

Identifies an operating or maintenance procedure, practice, condition, statement, etc, which if not strictly followed could result in equipment damage or destruction of, or serious impairment of system operation.

NOTE

Highlights certain operating or maintenance condition or statement which is essential but not of known hazardous nature as indicated by a warning or caution.

1-4 DESCRIPTION

The Fuel Rate of Flow Test Stand consists of two major assemblies and a test accessories set. The major assemblies are the test stand, built in two sections for shipment purposes, and the electrical control cabinet. The test accessories set comprises of hardware required during test stand operation. The test stand is designed to operate in a hazardous area, as defined by National Fire Protection Association (NFPA), using MIL-C-7024B type II test fluid to simulate the regular aircraft fuel.

The test stand consists of two sections bolted together to form an integral unit. Each section has a base which incorporates a drip pan. All major components, mounted in the test stand, are attached to the framework and are removable.

The electrical cabinet is designed to be air purged and is located adjacent to the test stand. All instruments, gages and switches are conveniently located with operator ease in mind. Forklift openings are provided in the base of the test stand.

1-5 LEADING PARTICULARS

The Fuel Rate of Flow Test Stand is built in two sections and the electrical control cabinet. One of the sections houses the reservoir, main pump, motor, etc. while the other section houses the Unit Under Test (UUT) area, main filter, etc. The leading particulars are as follows:

Dimensions:	
Test Stand Section (Reservoir, Pump)	
Height	72.00 inches
Width	138.00 inches
Depth	49.00 inches
Test Stand Section (UUT Area, Main Filter)	
Height	72.00 inches
Width	138.00 inches
Depth	53.00 inches
Electrical Control Cabinet	
Height	72.00 inches
Width	23.00 inches (at widest point)
Depth (Including Stop/Start Enclosure)	57.00 inches
Overall Test Stand Width	159.00 inches
Total Weight:	12,700 pounds (approximately)
Environmental Conditions:	
Operating temperature	60°F to 110°F
Humidity	90% at 110°F
Fuel System:	
Fluid used for testing	MIL-C-7024B TYPE II
Reservoir capacity	450 U.S. gallons
Test range	50 to 115,000 pounds per hour (PPH)
Accuracy	1% of set point or +10 PPH, whichever is higher
Level Gage: (Item No. 539)	Top mount, gear action, side-view fractional dial
Accuracy	+/-4%
Main Centrifugal Pump:	
Item No. 504	115,000 PPH (300 GPM) @ 110 PSIG @ 3,500 RPM

Main Motor:	
Item No. E601	30 HP, 460 VAC, 3-phase 60 Hz, 3,530 RPM
Services:	
Shop air	90-120 PSIG, 1/2" npt connection
Water supply	40 PSIG minimum pressure @ 60°F maximum temperature, 1-1/2" npt connection
Water return	20 PSIG maximum back pressure, 1-1/2" npt connection
Electrical Power:	
Main	480 VAC, 60 Hz, 3-phase
Controls and Instrumentation	115 VAC, 60 Hz, single phase
Low Frequency Square Wave Power Supply (Item No. E8)	Output: 0-75 VAC; 4-10 Hz, 2 or 3-phase
Output Current	2A from 10-18V; 0.5A from 19-75V
24V DC Power Supply (Item No. E535)	Input: 115 VAC Output: 24 VDC Accuracy: +/- 1V Output Current: 1.2 AMPS
400 Hz AC Power Supply: (Item No. E547)	Input: 115 VAC, 60 Hz Output: 115 VAC +/- 1 VAC, 400 +/- 0.2 Hz Current: 0.95 AMP
Drain and Vent	For drip pans, 1" npt connection
Connections:	Reservoir Drain, 2" connection Sink Vent, 4" connection
Air Pressure Gages:	
Air Inlet Filter Regulator (Item No. 521)	Set at 90 PSIG
Purge Pressure Regulator (Item No. 529)	Set at 10 PSIG
I/P Transducers Regulator (Item No. 543)	Set at 20 PSIG
Instrumentation:	
Voltmeter (Item Nos. E4 & E6)	0-199.9 VTRMS
Accuracy	0.5% of reading +/- 1 count

Ammeter (Item Nos. E5 & E7)	0-1.999 ATRMS
Accuracy	0.5% of reading +/- 1 count
Vibration Indicator (Item No. E12)	0-3 milli-inches (mills)
Accuracy	+/-0.5 mills
Inlet Pressure Indicator (Item No. E13)	0-75 PSIG
Press. Transducer (Item No. E531)	0-75 PSIG
Accuracy	+/-0.5%
Hourmeter (Item No. E14)	Records up to 99999.9 Hrs.
Temp. Controller (Item No. E15)	Input: Type 'T' Thermocouple; Output: 4-20mA
Diff. Press. Indicator (Item No. E19)	+30 to -30 PSID
Diff. Press. Transducer (Item No. E530)	+30 to -30 PSID
Accuracy	+/-0.25%
Back Press. Indicator (Item No. E20)	0-75 PSIG
Back Press. Transducer (Item No. E529)	0-75 PSIG
Accuracy	+/-0.5%
Audible Alarm (Item No. E22)	120 VAC System alarm
Voltmeter (Item No. E30)	0-19.99 VAC
Phase Angle Indicator (Item No. E31)	-179.995 to +179.995 degrees
Accuracy	+/- 0.001^{o}
Tracking Speed	1,000^{o}/Sec
Oscilloscope (Item No. E32)	
Bandwidth	DC to 20MHz
Sweep Rate	0.2 to 0.001 secs/division
Master Flow Indicator (Item No. E37)	
Interface	RS 232 Send Only
Low Flow Transducer (Item No. E523)	38-24,911 PPH (0.1-65GPM)
Accuracy	+/-0.1% of reading
High Flow Transducer (Item No. E524)	367-126,477 PPH (1-330GPM)
Accuracy	+/-0.1% of reading
Accelerometer (Item No. E510)	+/-2G Range, 12V Excitation
Accuracy	+/-5%

2nd Harmonic Converter (Item No. E545)	0 to 238°
Accuracy	+/- 0.5% of reading
Power	Self Powered
Universal Timer Counter (Item No. E546)	2 Channel, Input DC to 1MHz
Sensitivity	20mV
Interface	GPIB Full Control
Computer (Item No. E700)	IBM PC AT Compatible, 286-16 MHZ, 1.2 Meg floppy drive, 20 MB hard drive, MS DOS 3.3 AWARD BIOS
Operator Terminals #1, #2 (Item Nos. E38, E42)	Liquid crystal dot matrix type, 16 characters, display flash, 24 keys, RS-232-C interface, current 250mA max.

CHAPTER II

SPECIAL SERVICE TOOLS

2-1 SPECIAL TOOLS

No special tools are required for operation and maintenance of the test stand.

2-2 ACCESSORIES

A number of fittings, adapters, etc. are required to connect Unit(s) Under Test (UUT)s and test fixtures to the test stand. These are supplied with the test stand as a kit. The kit components are listed in Table 2-1. Included in the table are the component part number, Federal Supply Code for Manufacturers (FSCM) identification number and the quantity of each item in the kit. For adapters required with individual UUT, see Table 6-3.

Table 2-1. Kit Components List

COMPONENT	PART NUMBER	FSCM	QTY
Gaskets	40883-01	60984	2
Gaskets	40883-02	60984	2
Gaskets	40883-03	60984	2
Gaskets	40883-04	60984	4
Testing Vertical Fixture	40889	60984	1
Fluid Adapter	40999-01	60984	1
Fluid Adapter	40999-03	60984	1
Fluid Adapter	40999-05	60984	1
Fluid Adapter	40999-07	60984	1
Fluid Adapter	40999-11	60984	1
Fluid Adapter	40999-13	60984	1
Fluid Adapter	41004-01	60984	1
Cable Adapter	70018	60984	1
Cable Adapter	70019	60984	1
Cable Adapter	70020	60984	1
Cable Adapter	70036	60984	1
Cable Adapter	70037	60984	1
Cable Adapter	70044	60984	1
Cable Adapter	70045	60984	1
Cable Adapter	70046	60984	1

CHAPTER III

PREPARATION FOR USE, STORAGE AND SHIPMENT

3-1 PREPARATION FOR USE

The Fuel Rate of Flow Test Stand is shipped as a complete unit in two sections along with an electrical control cabinet. The unit is preserved and packaged to protect items during shipment and to provide protection from physical and mechanical damages. All open parts are sealed. The unit is covered with flexible material to exclude contamination and is mounted on shipping skids. Loose items which are part of the stand are packed in cartons and shipped with the stand.

3-2 INSTALLATION FACILITIES

The test stand is designed to operate in an area where the operating temperature is between 60 and 110°F (16 and 43°C). The test stand will be installed in hazardous Class I, Division 1, Group D location to pose a minimum risk to personnel and material. The facility should also have:

a. A floor which is level within +/- 2 degrees.

b. Sufficient lighting.

c. Sufficient ventilation.

d. An atmosphere as dust free as possible.

3-3 INSTALLATION REQUIREMENTS

Facility area requirements and service availability to the test stand are as follows:

a. Floor space area 138 inches wide and 102 inches deep with sufficient working space around the test stand (2 sections) to provide access for operation and maintenance.

b. Floor space area 48 inches wide by 23 inches deep for the electrical control cabinet (attached to UUT section) with sufficient working space around the cabinet.

c. Electrical power to the main motor of 480 VAC, 60 Hz, 3-phase, 50 amps and 115 VAC, 60 Hz, single phase, 40 amps for controls and instrumentation.

d. Shop air supply of 50 scfm at 90 to 120 psig.

e. Cold water supply of 40 psig minimum pressure at 60°F maximum temperature.

f. Cold water return of 20 psig maximum back pressure.

g. Drip pan drains.

h. Test stand electrical grounding.

i. Drain and fill provisions for the fuel reservoir.

3-4 UNPACKING AND LOCATING

The steps that follow are for removing the crating, inspecting the contents, and positioning and leveling the test stand.

CAUTION

Exercise care when unpacking the test stand to prevent damage to the finish, instruments or protruding parts.

a. Remove the wrapping material and examine the test stand for shipping damage as follows:

1. The frame is not bent, sprung or damaged.

2. No broken glass is found.

3. The gauges and instruments are not damaged.

4. All piping and tubing is undamaged and secure.

5. The valves and other controls operate freely without binding or excessive looseness.

6. Major equipment is securely mounted.

b. Locate the shipping document (packing list) and verify all items listed are accounted for.

c. Carefully remove and unpack all boxes; verifying the contents against the packing list.

d. Move the two test stand sections and electrical control cabinet to the permanent location. The rectangular slots built into the test stand section bases and electrical control cabinet base permit the use of a fork lift to raise the stand for removal of the shipping skids.

e. Level the test stand by holding levels at various points and installing shims wherever needed.

f. Remove all shipping stops (blocks, straps, holddown bolts, and packing) from the pump motors, pressure gauges, relays, motor starters and instruments.

g. Reconnect all matching connections which are numbered and tagged to ensure correct reinstallation between the two sections (Reservoir and UUT). Electrical control cabinet is already attached to the UUT section.

3-5 SERVICE CONNECTIONS

WARNING

MIL-C-7024B TYPE II calibration fluid causes skin irritation. Avoid prolonged or repeated contact. In case of skin contact, flush with water. Do not let it get in eyes. In case of contact with eyes, immediately flush with water for at least 15 minutes and call a physician.

Facility service connections are made at the rear and top of the test stand. Refer to engineering Drawing 10047, sheet 1 for the connection locations. (Drawing is Figure 11-1 and located in Chapter XI.) Table 3-1 lists the connections and the connector sizes.

Table 3-1. Service Connections

Item	Service Type	Connector Size
A	50 scfm at 90 to 120 psi shop air supply	1/2 inch npt
B	Cold water outlet	1-1/2 inch npt
C	Cold water inlet	1-1/2 inch npt
D	Drip pan drains (2)	1 inch npt
E	Reservoir drain	2 inch npt
F	Pressure fill	1 inch npt
G	115 VAC, 60 Hz, single phase, 40 amps facility power from customer to controls and instrumentation cabinet	1/2 inch npt conduit
H	480 VAC, 60 Hz, 3 PH power from customer to test stand main pump motor	1-1/2" npt conduit

3-6 ELECTRICAL CONNECTIONS

Facility electrical power of 480 VAC, 60 Hz, 3 phase to the main motor is supplied to the fuses of the motor starter while power of 115 VAC, 60 Hz, single phase to the electrical control cabinet is supplied to the terminal board of START/STOP box.

3-6-1 Connections Within The Test Stand Assembly

WARNING

All wiring and conduits to the test stand must conform to the NFPA National Electrical Code for hazardous areas. All electrical work must also conform to the NEMA standard code to protect personnel from injury due to explosion and/or fire.

a. The test stand is being shipped with wires and conduits interconnecting the section (UUT Area, Main Filter) and electrical control cabinet being connected. For connections between the two test stand sections, refer to matching tags and reconnect.

3-6-2 Connecting The Test Stand To The Facility's Supply.

WARNING

Ensure that facility power circuit breakers are open before connecting 480 and 115 volt circuits to the facility power.

Ground the test stand and the electrical cabinet to each other and to an external earth ground using a No. 6 AWG copper ground wire (minimum) connected to the grounding lug provided (see Figure 11-1).

NOTE

The external facility ground system used must be an absolute zero potential ground. A grounding lug is located on the base of the test stand at the rear for connection of ground.

a. Connect a wire from the facility's ground leg to the earth ground stud located at the rear bottom of the test stand.

b. Connect conduit to the main motor starter in Reservoir section.

c. Prepare to make power connections to the motor starter.

WARNING

Do not interchange a live lead with a ground lead. Such a mistake can cause serious injury. Be sure to identify the ground lead correctly. If a four-wire cable is used, the fourth conductor must be grounded to the motor starter.

d. Route the ground lead through the conduit, into the motor starter, and firmly secure it to the ground terminal.

e. Connect 480 VAC, 60 Hz, 3-phase facility power to terminals L1, L2 and L3, in the motor starter of the stand and 115 VAC, 60 Hz, single phase facility power to terminals H, N and GND in the START/STOP box outside the electrical control cabinet.

3-7 PHASE CHECKOUT

Before operating the test stand, check the phase relationship of the legs of the power supply.

CAUTION

The phasing determines the rotational direction of the motor. If the phasing is incorrect, then the pump will run backward. If the pump runs backward, it will be damaged. Such phasing needs to be corrected before operating the test stand.

To check the phasing, sample the rotation of main pump motor E601. Follow the procedure:

a. Perform steps 1 through 12 of paragraph 6-4.

b. On operator terminal #2, E42 select Test Code 24.

c. Turn MODE SELECT rotary switch E35 to AUTOMATIC or MANUAL position.

d. In quick succession, perform the two following steps while observing the rotational direction of motor E601:

e. Press PUMP/UUT POWER ON pushbutton E17 to turn motor on and immediately

f. Press PUMP/UUT POWER OFF pushbutton E16 to turn motor off.

g. Compare the direction observed to the direction indicated by the arrow affixed to the motor. If the directions are the same, then the phasing is correct. If not correct, continue this procedure with step h.

h. Reconfigure the supply connections as follows:

WARNING

Be sure to disconnect main power when changing leads. 480 Volt power can cause injury or death on contact.

NOTE

The ground wire is not one of the three conductors to be considered for reversing. Leave it as connected, and interchange any two of leads L1, L2, L3. One instance of reversing the leads is enough to correct the phasing. The stand has been phased-in at the factory.

i. Reverse two of the leads at the main terminals marked L1, L2, L3. Secure the leads as reconfigured.

3-8 STORAGE

If test stand has been in use, first follow procedures for normal shutdown as described in paragraph 6-5.

1. Ensure all supplies to the test stand at facility are turned off.

2. Disconnect all service connections given in table 3-1. Collect outflowing fluid in suitable, separate containers.

3. Unplug and empty all fluid from drip pan drain ports.

4. As much as practical, empty fluid from all filters and hoses.

5. Close all valves. Cap all test and service ports.

3-9 SHIPMENT

To prepare the test stand for shipment, first follow the storage instructions of paragraph 3-8. Then, disconnect the two sections and electrical control cabinet from each other.

Matching connections should be numbered and tagged properly to ensure correct reinstallation. Then, protect all instruments and controls with packaging in accordance with good shipping practices. Mount the sections on skids and protect them with suitable material.

CAUTION

To prevent equipment damage, shipment of test stand by rail is not recommended.

CHAPTER IV

FUNCTIONAL DESCRIPTION

4-1 GENERAL

This chapter contains the functional description of the Rate of Flow test stand systems, components and circuits. The stand includes Fuel, Pneumatic (including Purge Air), Water, Ventilation, Mechanical, Electrical and Instrumentation systems. Prior to and after repair, the fuel flow transmitters for various engines must be functionally tested and calibrated for conformance with the respective technical order requirements (see Table 1-1). The Fuel Rate of Flow test stand performs this function. The rated flow capacity of the test stand is 50 to 115,000 pounds per hour (PPH) using calibrating fluid MIL-C-7024B, Type II. All components are housed in two welded steel frames with access doors. Explosion proof electrical fittings and static free hoses are used throughout the test stand. A fuel flow transmitter to be tested is inserted in the transmitter clamping cylinder 541 (Figure 4-1, located at the end of the chapter) located on the right hand side of the unit under test (UUT) area. The clamping cylinder is bolted to a vibration plate mounted in a drip pan. The drip pan sink drain returns fuel spillage to the waste reservoir 501, from where it is pumped through low pressure filter 524 back to the fuel reservoir 500. An exhaust system removes test fuel fumes from test stand interior and UUT area. Appropriate test cables are connected between the transmitter under test and indicator receptacle E509 in the UUT area, which transmits the data to the electrical control panel.

4-2 THEORY OF OPERATION

An electric motor E601 drives pump 504 (Figure 4-1) and moves the test fuel from the reservoir 500. The fuel then passes through a computer E700 selected air operated shut-off valve 511, 512 or 513 and through the respective flow rate controller 514, 515 or 516 to the three way diverter valve 517. The diverter valve allows the fuel to flow either through the flow transducers E523 and E524 or bypasses and returns the fuel back to the reservoir 500. If the transmitter being tested is ready for test, the diverter valve 517 is normally set to allow fuel to flow through the flow transducers E523 and E524. Test fuel flow is held to a predetermined rate, as directed by the computer E700, permitting flow from flow transducers to the transmitter under test in UUT area. The test stand operator records the necessary indicator readings from the control chassis of electrical control cabinet.

4-3 FUEL SYSTEM

The Fuel Rate of Flow test stand fuel system consists of a self-contained recirculating system with a centrifugal pump and explosion proof motor, fuel reservoir, flow control valves, automatic temperature control, filter, flow transducers and fuel return circuit. The following paragraphs describe the functional operation of the major components and circuits (Figure 4-1).

4-3-1 Fuel Reservoir

The stainless steel fuel reservoir 500 has a fluid capacity of approximately 450 gallons. During normal operations, with a test transmitter connected to the test stand using test kit hardware, fuel flows from the reservoir 500 through the system and returns to the reservoir at controlled temperature and flow rate. The fuel reservoir is normally filled by pressure through the ball valve 523 and low pressure filter 524. The pressure fill connection is located at the rear of test stand. The reservoir may be manually filled through the safety fill cap 525 also. The reservoir is also replenished from waste reservoir 501, when it fills up, by centrifugal pump 527 through the low pressure filter 524. Fluid level is displayed on level gage 539. Fuel level is also monitored by main fuel reservoir level switch E515. When the fuel level falls below 1/2 full, the level switch contacts open and shut down the power supply to main motor E601. This also lights up the red system fault indicator light E21 along with activating the audible alarm E22 and maintains alarm messages on operator terminal #2, E42. The reservoir is vented to atmosphere through a hooded flame arrestor 526 mounted on top of the reservoir. Fuel temperature in reservoir is monitored by fuel reservoir temperature switch E514. When the temperature in reservoir rises to 90 +/- 2^{O}F, the temperature switch contacts open and shut down the power supply to main motor E601. This will light up the red system fault indicator light E21, along with activating the audible alarm E22 and maintains alarm messages on operator terminal #2, E42. Drain valve 530 allows the reservoir to be drained. Drain valve is normally held in the closed position. It also is connected to the fusible link 532 mechanism. In the event of fire or extremely high temperatures, the fusible link will melt, thereby allowing the weighted drain valve 530 to open and drain the reservoir 500. Baffles are provided in the reservoir to prevent pressure pulsations and minimize turbulence and aeration. Fuel is supplied from the reservoir through shut-off valve 502 to the inlet side of centrifugal pump 504.

4-3-2 Fuel Supply Circuit

The system start switch E2 is pressed and after satisfying the purge flow requirements, power is automatically applied to the control electronics. When the conditions of air pressure, water pressure, fuel temperature inside reservoir 500 and in line to UUT, and level in reservoir 500 are satisfied, the centrifugal pump 504 can be started by pressing the PUMP/UUT Power On switch E17. The output of the pump is routed through check valve 505 and 10 microns main filter 508 to the air operated shut-off valves 511, 512 and 513. The heat exchanger 506 is capable of automatically controlling test fuel temperature in the reservoir 500 to 80 +/- 10°F through the use of water circulation and a temperature control water regulating valve 507. Flow pulsations are dampened by the accumulator 509, which is precharged to 25 PSIG with Nitrogen gas. Relief valve 510 maintains system pressure at 85 PSIG, and diverts excess pump flow back to the reservoir 500. According to the test parameters selected, the computer E700 sends signal to open one of the flow solenoid valves E538, E539 or E540 which opens its respective air operated ball valve 511, 512 or 513. The fuel flows accordingly through one of the flow rate controllers 514 (40-590 PPH), 515 (400-7,700 PPH) or 516 (5,800-115,00 PPH) to the three way diverter valve 517. For normal testing situation, the diverter valve allows the flow to the flow transducers E523 and E524. For response test preset, the diverter valve closes the flow through flow transducers and diverts the fuel flow to back pressure regulator valves 519 and 520. Also as per test parameters selected, the computer E700 sends a signal to master flow select solenoid valve E537 to close air operated shut-off valve 518 if low rate of flow (50-25,000 PPH) transmitters are being tested, forcing all UUT flow through E523, the low flow master flowmeter. If high rate of flow (350-120,000) transmitters are being tested, valve 518 is open and the high flow master flowmeter E524 measures UUT flow. Two of the three needle valves 535 are normally open, to read inlet pressure, back pressure and differential pressure readings, while the third needle valve 535 should be closed. The third needle valve is opened only to take fuel samples or for purging purposes.

4-3-3 Fuel Return Circuit

Fuel used in testing returns to fuel reservoir 500 through check valve 538 and back pressure regulator valves 519 and 520. To relieve high back pressure, a dump valve 536 is installed in the return circuit.

4-3-4 Waste Fuel Circuit

All the fuel that drips or is purged in sink at UUT area, flows to the waste reservoir 501. When the fuel level rises in the waste reservoir to 2/3 full level, the residue reservoir level switch E507 makes contact and energizes residue transfer pump solenoid E508 allowing the air to flow to air motor 528. The air motor turns, driving the centrifugal pump 527 which takes fuel suction from waste reservoir 501 and pumps it to the reservoir 500 through the check valve 533 and low pressure filter

524. A sight glass 534 on the waste reservoir 501 shows the level of fuel in the waste reservoir. After some time, the residue level switch reaches the open contact position and then starts the relay timer E520. The relay timer has a delay on release set for 30 seconds. This allows the pump to pump out fuel for an additional 30 seconds to reduce the reservoir level. At the end of 30 seconds, the relay timer E520 deenergizes the residue transfer pump solenoid E508, thereby stopping the air motor 528. The relay timer could be adjusted between 1 - 60 seconds for holding on of waste fuel pumping and it should be so adjusted that at the end, approximately 1/3 full level remains in the waste reservoir.

4-4 PNEUMATIC SYSTEM

The test stand pneumatic system takes the shop air supply (90 to 120 PSIG) and distributes it to the various pneumatic circuits (Figure 4-1) as described below. Shop air enters the test stand at the rear. For test stand to function, shop air supply must be applied at all times. When the System Start pushbutton E2 is pressed, it energizes and opens the air solenoid valve E504. This allows the air to pass through to filter/regulator 521 where it is regulated to 90 PSIG. It passes through coalescing filter 522 where water and impurities are trapped and filtered air at 5 microns passes on to the rest of the pneumatic circuits. An air pressure switch E513, is set at 70 PSIG. Should the air pressure at this point drop below 70 PSIG, main pump operation is inhibited. The air is now available at residue transfer pump solenoid valve E508, flow solenoid valves E538, E539 and E540, bypass solenoid E536, master flow select solenoid valve E537, positioners of back pressure regulators, four way air valve 542 for clamping cylinder, purge air pressure regulator 529 and I/P transducers pressure regulator 543. As required by the test parameters, the computer E700 selects the appropriate solenoids and valves mentioned above.

4-4-1 Residue Reservoir Pump Circuit

When residue reservoir level switch E507 closes, it energizes the residue transfer pump solenoid E508. The solenoid valve opens and allows the shop air to turn the air motor 528, which drives the centrifugal pump 527. When residue level switch opens, it starts a delay on timer E520 and when that times out, it deenergizes the solenoid valve E508, stopping the air supply to the air motor.

4-4-2 Purge Air Circuit

Purge air is provided for purging the electrical cabinet. Shop air is reduced by the pressure regulator 529. Pressure at the regulator is set to whatever pressure is necessary to maintain 0.15" W.C. in the purged enclosure. Pressing the System Start pushbutton E2 allows the air to flow through the air amplifier 540 and into the electrical cabinet. When purge pressure is maintained, the purge timing relay E505 comes on. The timing relay times out after approximately 2 minutes of powering up of test stand. A purge relief valve 553 is attached to the cabinet and will relieve pressure exceeding 0.15 inch of water. A purge pressure differential switch E503 set at 0.15 inch of water is placed in the inlet line of purge air to electrical enclosure. Should the differential pressure of purged air drop below 0.15 inch of water, switch E503 would open and shut down power to the entire test stand, preventing hazardous conditions. This will also illuminate purge fail light E1.

4-4-3 Flow Circuit

In accordance with the test parameters selected, the computer E700 regulates and selects the correct valves. Depending upon the amount of flow required, it energizes one of the three flow solenoid valves E538, E539 or E540. Shop regulated air flows through one of these valves to the respective air operated shut-off valve 511, 512 or 513. This places the valve in open position allowing the fuel to flow to the respective fuel flow rate controller 514, 515 or 516. The fuel flow rate controller regulators also receive an electrical signal from the computer E700 and regulate the amount of flow rate through the respective controllers. The diverter valve 517 is normally set to allow fuel flow from controller to the flow transducers E523 and E524. When response test is performed, then the diverter flow to flow transducer is quickly diverted around the UUT. This is achieved by pressing stop flow pushbutton E40 which sends signals to energize bypass solenoid E536 and allows shop air through quick exhaust valve 537 which places the diverter valve in the required position. Also depending upon the rate of flow selected, the master flow select solenoid valve E537 maybe energized. Normally the flow is set up to go directly through high flow transducer E524. When low flow is required, the master flow select solenoid valve energizes and allows shop air to activate air operated shut-off valve 518. The fuel then passes through low flow transducer E523 and high flow transducer E524 to the UUT. Upon deenergizing of master flow select solenoid E537, the air flow to air operated shut-off valve 518 shuts off. Shop air also supplies to four way air valve 542 to position and clamp the UUT.

4-4-4 I/P Transducers Circuit

Shop air flows to the pressure regulator 543 where it is regulated down to 20 PSIG. This air is supplied to I/P transducers E525 and E533.

4-5 WATER SYSTEM

Facility cooling water supplied to the test stand service at 40 PSIG pressure minimum and 60°F maximum should be used for maintaining the fuel temperature within required limits (80 +/- 10°F). Water enters the test stand through shut-off valve 531 at the rear of stand. When valve 531 is opened, pressure switch E512 monitors water pressure and disables pump operation should pressure fall below 40 PSIG. Water enters plate type heat exchanger 506 and cools fuel in the reservoir 500. A fuel temperature thermocouple E522 monitors fuel temperature downstream of the centrifugal pump 504 and sends signal to the panel mounted temperature controller E15. It sends the signal to the I/P transducer E533 which sends the control signal to the diaphragm of water regulating valve 507. The water regulating valve is located in the water return line from the heat exchanger. The signal from the thermocouple translated to the regulating valve automatically regulates water flow through the heat exchanger to maintain fuel temperature of approximately 80°F. Water returns to the facility system through a service panel port located at the rear of the test stand.

4-6 VENTILATION SYSTEM

The test stand is equipped with an exhaust system consisting of a motor, blower and ducting. Ducting is provided for exhausting fumes and vapors from the sink area and from each section of the test stand. Ducting terminates in a 4-inch diameter duct which extends through the roof of the test stand for connection to the facility exhaust system.

4-7 MECHANICAL SYSTEM

A mechanical system is provided within the test stand to mount and connect all UUT transmitters, to supply 60 Hz vibration during testing and fusible link assembly (Figure 4-1).

4-7-1 Transmitter Clamp Assembly

The transmitter clamp assembly provides for mounting fuel flow transmitters to be tested. The clamp assembly consists of a stationary mount, bolted to a vibration plate, and a movable clamping cylinder 541 also bolted on the vibration plate. The clamping cylinder is air operated by 4-way air valve 542 and is adjustable to accommodate the various types of fuel flow transmitters. Combination of various adapters and clamping cylinder secure the UUT(s) in position. The transmitter clamp assembly and vibration plate are shock mounted in a sink which collects and returns fuel spillage to the waste reservoir 501. A shipping tube sealed with rubber gaskets is provided to complete the fuel flow circuit when a transmitter is not in test position.

4-7-2 Vibration Plate

The vibration plate supports the transmitter clamp assembly. An electrically operated vibrator E511, attached to the vibration plate, in combination with accelerometer E510 and variable transformer E9, serves to vibrate the transmitter clamp assembly. Panel mounted adjustable control switch E9 allows the amplitude of vibrations desired.

4-7-3 Fusible Link Assembly

Fusible link assembly consists of wires mounted over pulleys attached to fusible link 532, drain valve 530 and supporting roller of fusible link sensor limit switch E548-4. One fusible link is located in a section of test stand over main pump and the other is placed in sink area. If temperatures in sink area or inside the test stand should exceed 160°F, the fusible link(s) will melt, thereby releasing the wire and allowing the weighted drain valve 530 lever to place the valve in open position. This will allow the fuel in the reservoir 500 to be drained into the facility connected drain line. The wire tension holds the roller of fusible link sensor limit switch E548-4 in such a way that the limit switch is in closed position, thereby completing electrical path and energizing the test stand. When the fusible link(s) melt, the wire loses tension allowing the fusible link sensor limit switch to open. This enables the test stand power to shut off, thereby preventing any further hazardous conditions.

4-8 ELECTRICAL SYSTEM

Electrical power of 480VAC, 60Hz, 3-phase will be used for operation of the main pump motor E600 (Figure 4-2, located at the end of the chapter) in the test stand. Electrical power of 115VAC, 60Hz, single phase is supplied to the electrical cabinet from facility and is used to power all controls and instrumentation and special AC voltages and frequencies. DC voltages are obtained from power supplies or generated by specialized circuits within the test stand.

4-8-1 Lighting Circuits

115VAC power is available for lighting circuits and flows through fuse E552. Should any of the normally used access doors be opened in both the sections of the test stand, they release the rollers of door interlock limit switches E548-1, E548-2 or E548-3. This closes the contact in the respective limit switch and turns on the maintenance light E549. Closing the access door(s) shuts off the maintenance light automatically. Turning the light switch E554, located in sink area, to the ON-position turns on the sink light E553. Turning the light switch to OFF position turns off the sink light.

4-8-2 Start/Stop Circuit

115VAC power flows through fuse E500 to the normally closed Emergency Stop switches E50 and E23, fusible link sensor limit switch E548-4 and System Stop switch E3 to the System Start switch E2. Pressing E2 energizes System Start/Vent Fan Relay E502 which closes all its normally open contacts and also turns on the green system on light

E2. One of the E502 contacts maintains holding contact to maintain flow once the pushbutton E2 is released. The other two E502 contacts allow current to flow and turn on the vent motor E600. This ensures that all vapors are constantly removed from both the test stand sections and the sink area when the System Start pushbutton is pressed. The current also energizes the air solenoid E504, allowing the facility air into the pneumatic system and start the purge air process. The red purge fail light E1 comes on as long as there is no purge pressure. Upon successful purge pressure in the electrical cabinet, purge pressure differential switch E503 turns off the purge fail light E1 and applies power to the purge timeout delay relay E505. When the timer E505 times out after 2 to 10 minutes, it applies power to the control and instrumentation system.

4-8-3 Controls and Instrumentation Circuit

Before centrifugal pump 504 can be started successfully the following conditions must be satisfied: 70 PSIG air pressure should be available to close air pressure switch E513, 40 PSIG water pressure should be available to close water pressure switch E512, temperature controller E15 should sense UUT fuel temperature to be less than 90 +/- 2°F, fuel temperature in reservoir 500 should be below 90 +/- 2°F to maintain fuel temperature switch E514 closed and fuel level in reservoir should be at least 1/2 full to close fuel level switch E515. Upon the above conditions being satisfied, malfunction relay E544 energizes. This closes its contact allowing the pump start circuit to be completed. Should any of the above conditions not be satisfied, computer E700 sounds the audible alarm E22, lights up the red system fault indicator light E21 and presents a message on terminal #2, E42. If the vent fan motor E600 does not come on or fails during the test, the vent flow switch E541 closes sounding the audible alarm E22, turns on the fault light E21 and flashes a message on terminal #2, E42 through the computer. But the UUT test can be completed and the pump motor started, inspite of the vent fan not working as this does not prevent the malfunction relay E544 from being energized.

After all above conditions have been satisfied, pressing the PUMP/UUT POWER ON pushbutton E17 allows the current to flow through closed PUMP/UUT POWER OFF pushbutton E16 and the energized malfunction relay E544 contact to the normally closed overload heater contact E517 and also energizes pump contactor E516. One E516 contact maintains holding circuit so that when PUMP/UUT POWER ON pushbutton E17 is released, it maintains the circuit and allows the pump 504 to run. This also turns on the PUMP/UUT ON indicator light E18 and energizes UUT isolation relays E518 and E519. These relays apply UUT power to the UUT test connector E509 only while the pump is on. Pressing the PUMP/UUT POWER OFF pushbutton E16 interrupts the pump circuit and stops the pump.

4-8-4 Residue Reservoir Circuit

The centrifugal pump 527 automatically starts and stops the fuel pumping from waste reservoir 501 to the main reservoir 500. When waste reservoir level rises to approximately 2/3 full, the residue level switch E507 makes contact and this energizes residue transfer pump solenoid E508. This permits air to flow and turn the air motor 528 which drives the centrifugal pump 527. When the residue level falls to about 1/3 full, the residue level switch E507 contact opens. This turns on the timer relay E520 which delays the release of solenoid E508 for approximately 30 seconds, allowing the remaining fuel to be pumped out. At the end of time delay, the solenoid E508 deenergizes closing the air supply to air motor and thereby stopping the pump 527.

4-8-5 Vibration Circuit

Turning the vibration on selector switch E10 to ON position, turns on the test table vibrator E511 and lights the vibration on indicator light E11. The amplitude of vibration can be adjusted by means of variable transformer switch E9. The vibrator E511 is also mechanically connected to the accelerometer E510.

4-9 INSTRUMENTATION SYSTEM

The instrumentation system includes voltmeters, ammeters, hourmeter, temperature controller, master flow indicator, pressure indicators, flow transducers, pressure transducers, power supply selector, phase angle indicator, oscilloscope, operator terminals and a system control computer along with its interfaces, digital and analog racks and cards (Figure 4-2).

4-9-1 Vibration Indicator

Vibration indicator E12 is a voltmeter which receives signals from accelerometer E510 and potentiometer E12-1 and reads out in milli-inches (mills).

4-9-2 Hourmeter

Hourmeter E14 reads out the hours that main pump 504 has been running.

4-9-3 Temperature Controller

Temperature controller E15 receives and reads out the reading from thermocouple E522. It is set to maintain 80°F operating temperature of fluid flowing to UUT. The temperature is maintained by sending signals to cooling water I/P transducer E533 which controls the water regulating valve 507 to maintain the set temperature. If the temperature goes to 90 +/- 2°F, the computer E700 sends signals to the fault indicator light E21 to light up, the audible alarm E22 sounds, message is flashed on terminal #2, E42 and the pump motor circuit is interrupted and shuts down the pump. The temperature controller can be programmed to set different parameters as required.

4-9-4 Master Flow Indicator

Master flow indicator E37 interfaces with the computer E700 through the RS232 interface card E700-3. The computer signals and selects to use low flow transducer E523 or high flow transducer E524 as dictated by flow select relay E521. The master flow indicator displays the flow in PPH after compensation for temperature information received from temperature transducer E524-1 has been computed by the computer.

4-9-5 Pressure Indicators

Differential pressure indicator E19, back pressure indicator E20 and inlet pressure indicator E13 also interface with the computer E700 through the RS232 interface card E700-3 and the analog I/O adapter E701. Differential pressure indicator receives signal from differential pressure transducer E530 and displays pressure in PSID. Back pressure indicator and inlet pressure indicator receive signals from their respective back pressure transducer E529 and inlet pressure transducer E531 and display pressures in PSIG.

4-9-6 Linear Servo Voltmeter

Linear servo voltmeter E30 receives linear servo signal from the UUT and displays values in ac volts.

4-9-7 Low Frequency Power Supply

Low frequency power supply E8 provides various frequencies, phases and voltages required for UUT tests. The frequencies and phases which are available are: 4 Hz 3-phase, 6 Hz 2-phase, 6.6 Hz 2-phase and 8 Hz 2-phase. These can be selected by means of a selector switch. Voltage is continuously adjustable from the front panel.

4-9-8 Low Frequency Voltage and Current

Low frequency volts true root mean square (VTRMS) and low frequency amperes true root mean square (ATRMS) supplied by the power supply E8 to the UUT is read on respective voltmeter E6 and ammeter E7. The voltmeter and ammeter also interface with the computer E700 through the analog I/O adapter E701 and RS232 interface card E700-3.

4-9-9 400 Hz Voltage and Current

400 Hz VTRMS voltage and 400 Hz ATRMS supplied by power supply E547 to the UUT is read on respective voltmeter E4 and ammeter E5. The voltmeter and ammeter also interface with the computer E700 through the analog I/O

adapter E701 and RS232 interface card E700-3.

NOTE

400 Hz UUT power is not adjustable from the front panel. 115VAC or 26VAC is automatically selected by jumpers within the appropriate UUT interface cable.

4-9-10 Phase Angle Indicator

Phase angle indicator E31 receives the signal from the UUT and sends the information to computer E700 through interface card E700-4. The phase angle indicator displays the phase angle from UUT in degrees and its range is between +180° to -180°. The computer converts this angle to 0-360 degrees.

4-9-11 Oscilloscope

Transient monitor oscilloscope E32 receives its signals from either inlet pressure signal from any transducer or from drum or pickoff signals from the 9-196-01 UUT. The signal desired can be selected by means of scope select switch E36. The oscilloscope displays wave shape vs:time on its monitor.

4-9-12 Operator Terminals

Operator terminal #1, E38, and operator terminal #2, E42, interface with the computer E700 through the RS232 interface card E700-3. These terminals communicate to and from with the computer, act as keyboard for the computer, display messages, lead the operator through the test parameters for the selected UUT, display flow, differential pressure, angle indicator readings and any other system messages. When key switch E41 is placed in the PROGRAM mode, the operator terminals can add or change the programs for the desired UUT by inputting the data from its keys.

4-9-13 Universal Timer/Counter

Universal timer/counter E546 receives signals from the UUT. It calculates drum and pickoff timing intervals to determine UUT angle and displays the same on operator terminal #1, E38.

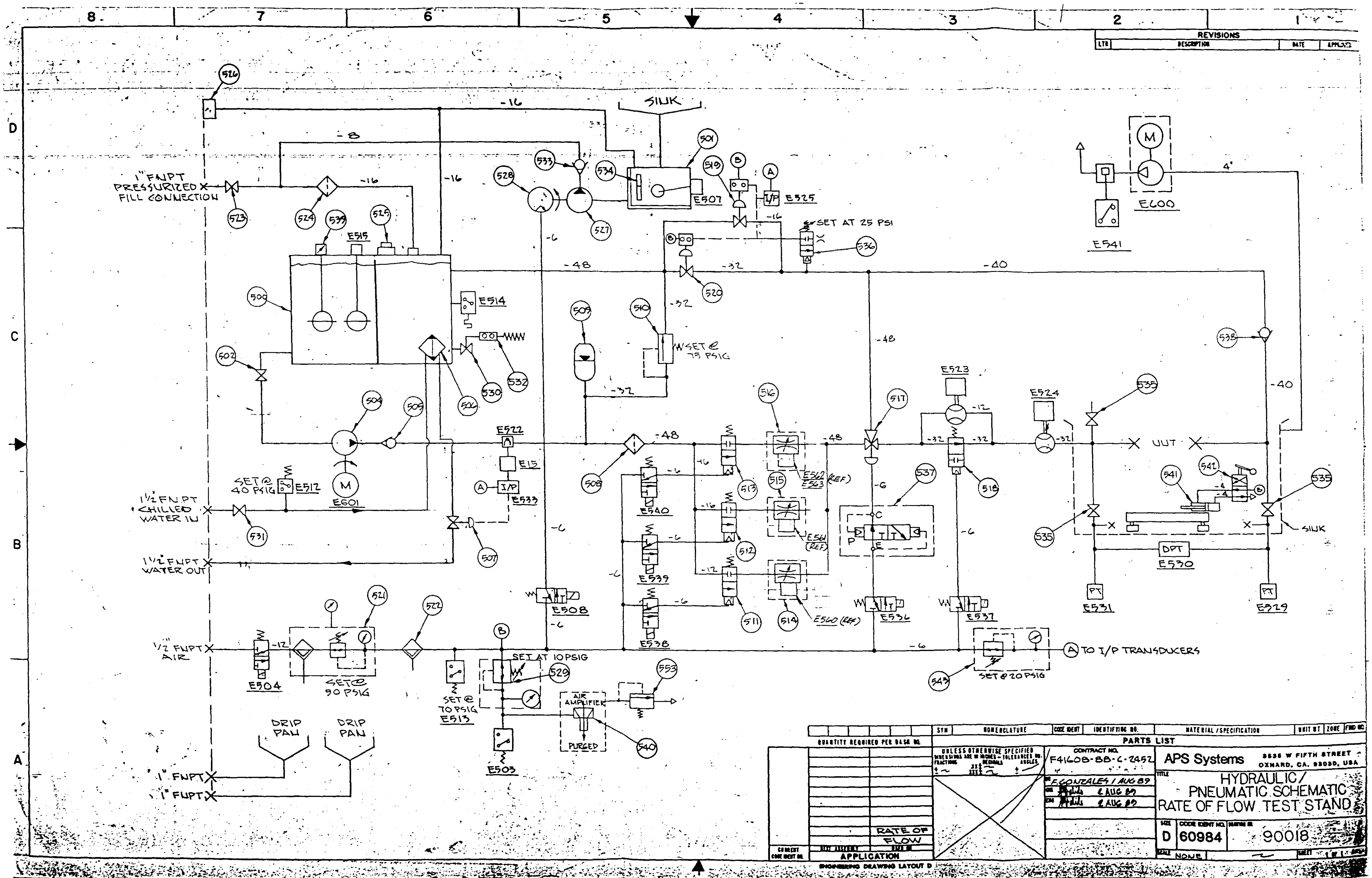


Figure 4-1. Hydraulic and Pneumatic Schematic

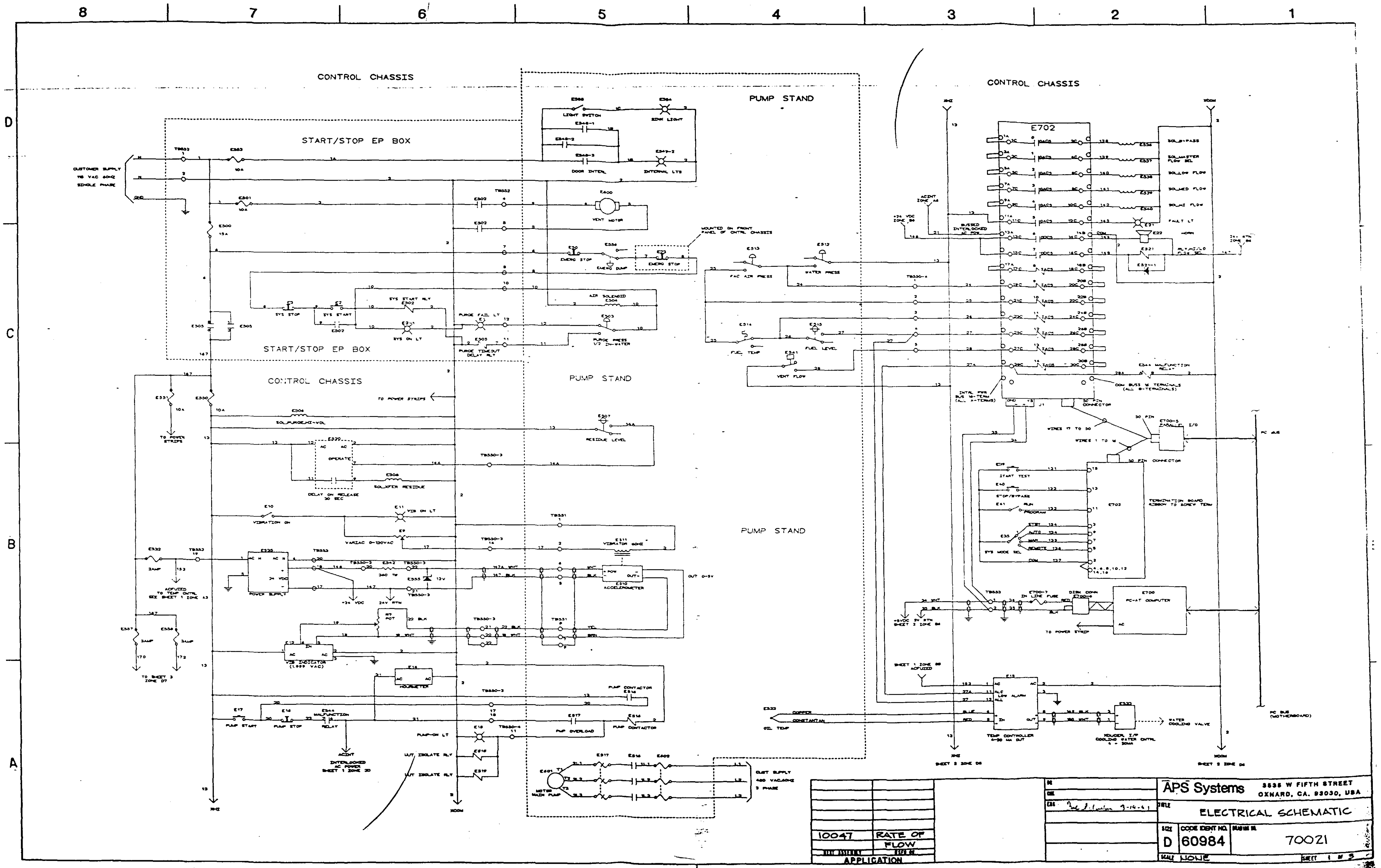


Figure 4-2. Electrical Schematic (Sheet 1 of 3)

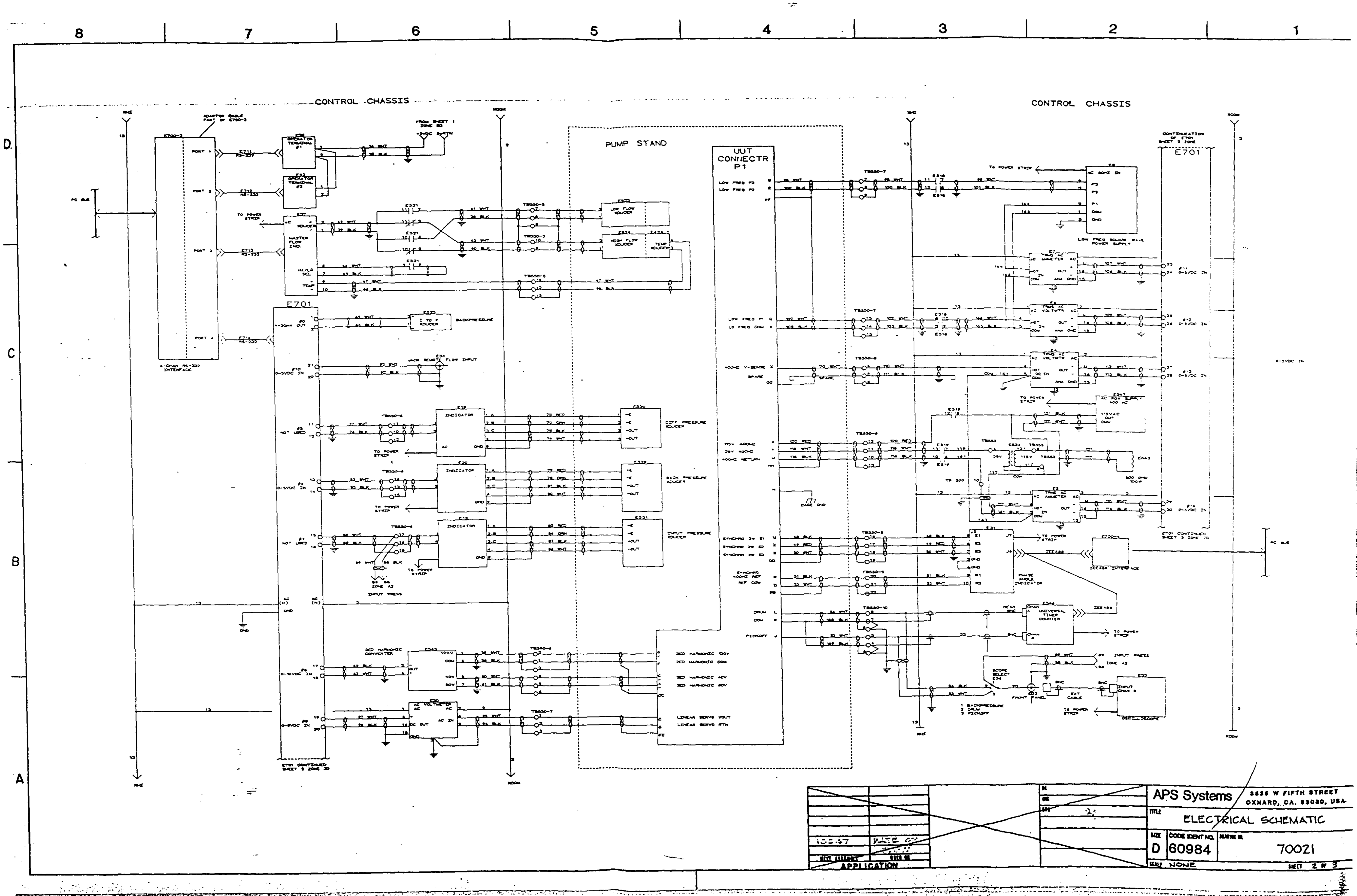


Figure 4-2. Electrical Schematic (Sheet 2 of 3)

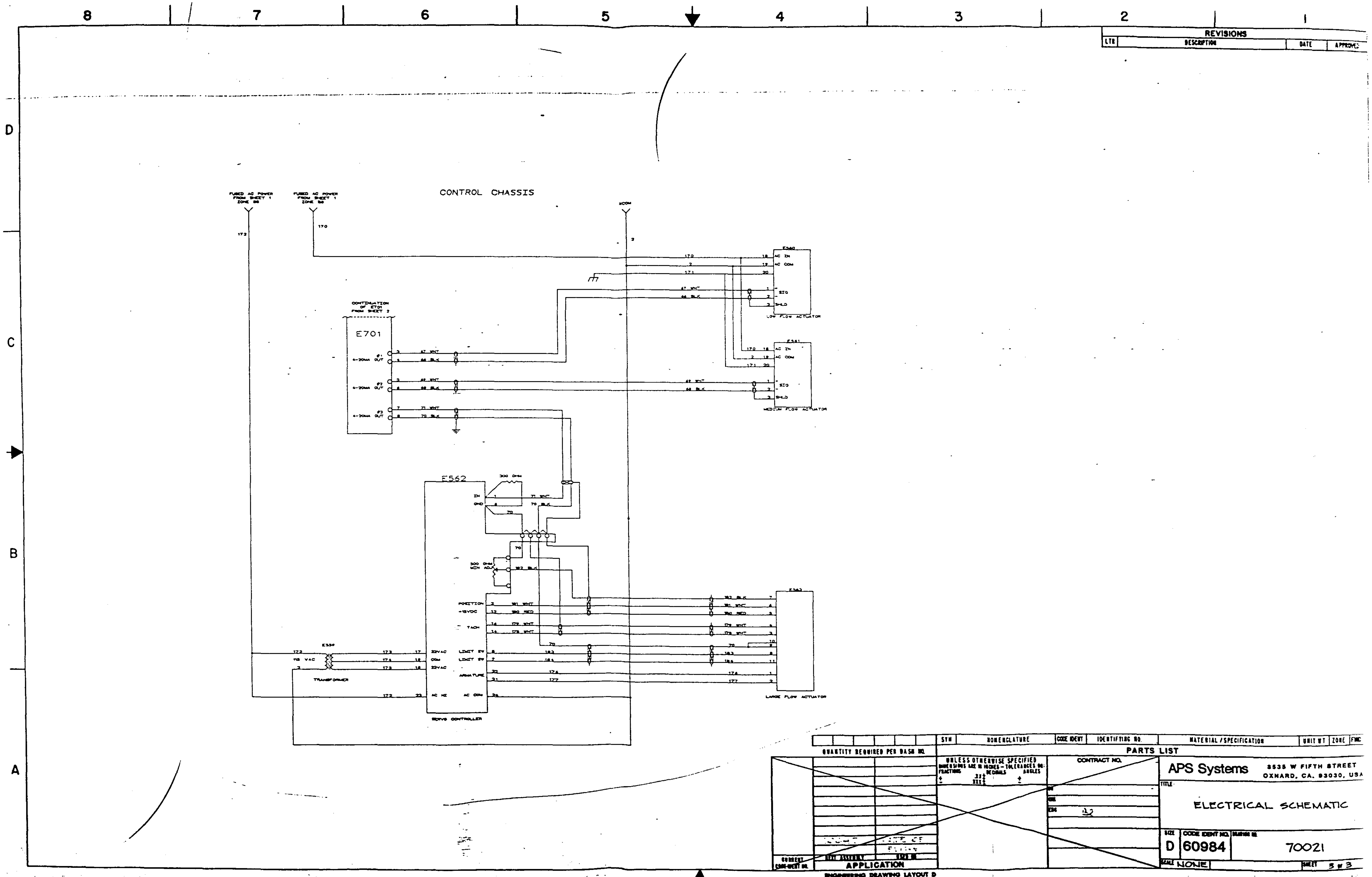


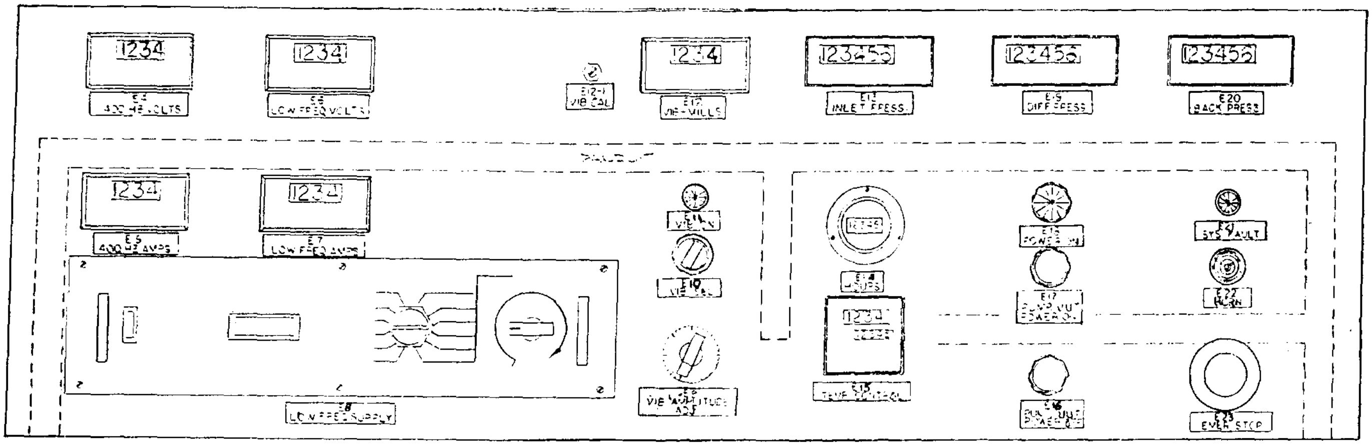
Figure 4-2. Electrical Schematic (Sheet 3 of 3)

CHAPTER V

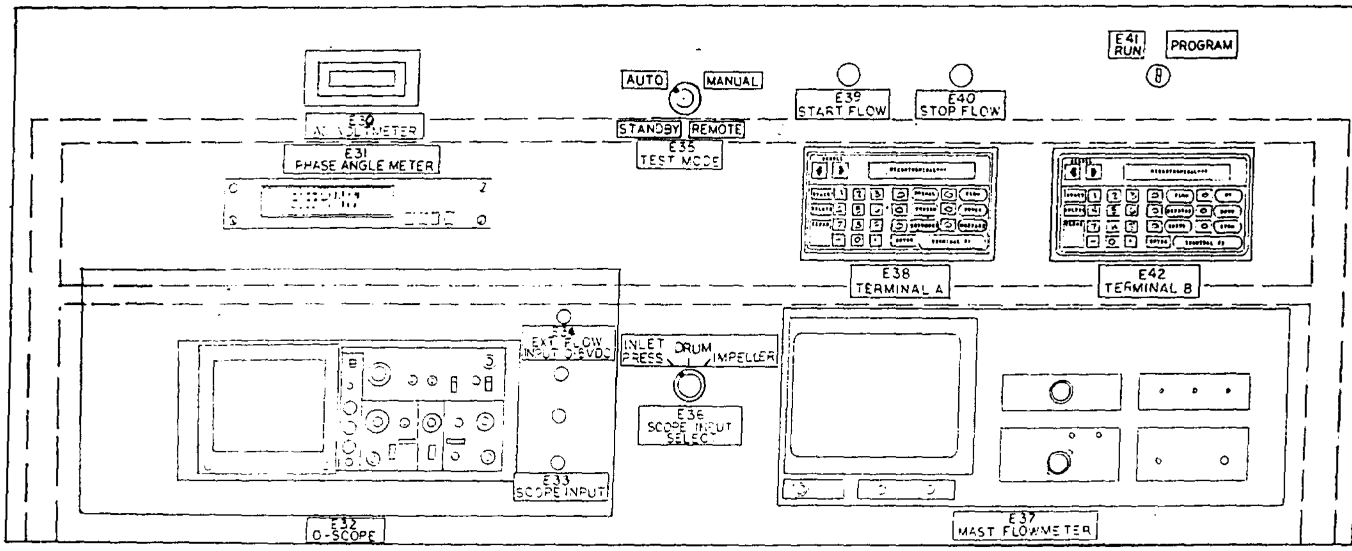
DESCRIPTION OF CONTROLS AND INSTRUMENTS

5-1 GENERAL

This chapter lists the operating controls and indicating instruments used on the test stand. Each entry cites the item number and description of the component along with its function (Tables 5-1 and 5-2). Careful study of each operating control and indicating instrument will increase operating efficiency. The majority of the electrical controls and instruments are located on the electrical control cabinet panels (Figure 5-1). Electrical controls and instruments located elsewhere have their locations indicated in the description column. Operator controlled valves, filters, etc are shown in Figure 5-2 and details are given in Table 5-2.



Upper Panel
Figure 5-1. Test Stand Electrical Controls and Instruments
(Sheet 1 of 2)



Lower Panel
Figure 5-1. Test Stand Electrical Controls and Instruments
(Sheet 2 of 2)

Table 5-1. Electrical Controls and Instruments

ITEM NO	DESCRIPTION	FUNCTION
E1	PURGE FAIL Red Indicator Light (On start/stop box)	When light is off, it indicates positive purge pressure in electrical control cabinet
E2	SYSTEM START Pushbutton with Green Pilot Light (On start/stop box)	Pressing the pushbutton energizes the SYSTEM START/VENT FAN relay E502, starts vent motor E600, energizes air solenoid relay E504, turns on the green light E2 and red light E1 and starts purge cycle. System automatically powers on in approximately 2 minutes if purge uninterrupted and then red light E1 extinguishes.
E3	SYSTEM STOP Pushbutton (On start/stop box)	Pressing the pushbutton interrupts the system start circuit, deenergizes the SYSTEM START/VENT FAN relay E502, stops vent fan motor E600, deenergizes air solenoid relay E504 and turns off the green light E2 and turn on the red light E1. Resets purge timeout to zero.
E4	400 Hz Voltmeter	Displays value in volts true root mean square (VTRMS) from unit under test (UUT) on voltmeter and also provides this information to computer E700.
E5	400 Hz Ammeter	Displays value in amperes true root mean square (ATRMS) from UUT on ammeter and also provides this information to computer E700.
E6	LOW FREQUENCY Voltmeter	Displays value in VTRMS from UUT on voltmeter and also provides this information to computer E700.
E7	LOW FREQUENCY Ammeter	Displays value in ATRMS from UUT on ammeter and also provides this information to computer E700.

Table 5-1. Electrical Controls and Instruments - Continued

ITEM NO	DESCRIPTION	FUNCTION
E8	LOW FREQUENCY Power Supply	Pressing the switch to ON position allows low frequency power to UUT. A selector switch allows to choose from: 4 Hz 3-phase, 6 Hz 2-phase, 6.6 Hz 2-phase and 8 Hz 2-phase. A voltage adjustment switch allows to choose voltage between 0 to 75 VAC and displays the voltage value.
E9	VIBRATION AMPLITUDE ADJUSTMENT	Turning the adjustment switch allows to select the amplitude of vibrations desired.
E10	VIBRATION Selector Switch	Turns on/shuts off vibration system of test stand.
E11	VIBRATION ON Green Indicator Light	Illuminated light indicates vibrator E511 is on.
E12	VIBRATION 'MILLS' Voltmeter	Displays vibration readings in mills.
E12-1	VIBRATION CAL Potentiometer	Used for calibrating vibration voltmeter E12.
E13	INLET PRESSURE Indicator	Displays value of inlet pressure in PSIG received from inlet pressure transducer and also transmits it to computer E700.
E14	PUMP HOURS Hourmeter	Indicates total time that main pump 504 has been in use.
E15	FUEL TEMPERATURE Controller	Programmed to achieve fluid temperature desired. Usually set to 80 Deg F. Receives and displays reading from thermocouple E522. Sends signal to I/P transducer E533 to regulate valve 507 accordingly. When fluid temperature goes to 90 +/- 2 Deg F, sends signal to computer E700 which disables pump operation.

Table 5-1. Electrical Controls and Instruments - Continued

ITEM NO	DESCRIPTION	FUNCTION
E16	PUMP/UUT POWER OFF Pushbutton	Pressing the pushbutton interrupts pump motor E601 circuit; thereby stopping the motor. Also shuts off power at UUT test connector E509.
E17	PUMP/UUT POWER ON Pushbutton	Pressing the pushbutton energizes the pump 504 circuit, if all other pressure, temperature and level conditions have been satisfied. This enables motor E601 and pump 504 to start. Also power is available at UUT test connector E509.
E18	PUMP/UUT POWER ON Amber Indicator Light	Illuminated light indicates pump/UUT power is on.
E19	DIFFERENTIAL PRESSURE Indicator	Displays value of differential pressure in PSID received from differential pressure transducer and also transmits it to computer E700.
E20	BACK PRESSURE Indicator	Displays value of back pressure in PSIG received from back pressure transducer and also transmits it to computer E700.
E21	SYSTEM FAULT Red Indicator Light	Illuminated light indicates the following conditions may not be satisfactory: Shop air pressure, facility water pressure, fuel reservoir temperature, fuel reservoir level, vent flow, UUT fuel temperature, GPIB or RS232 faults.
E22	SYSTEM ALARM	Audible alarm sounds if any of the conditions listed under E21 are not satisfactory.
E23	EMERGENCY STOP Pushbutton	Enables shutdown of control power to test stand.

Table 5-1. Electrical Controls and Instruments - Continued

ITEM NO	DESCRIPTION	FUNCTION
E30	LINEAR SERVO Voltmeter	Displays value of linear servo signal received from UUT in AC VOLTS.
E31	PHASE ANGLE Indicator	Displays value of phase angle from synchro type UUT in DEGREES and also transmits it to computer E700.
E32	TRANSIENT MONITOR OSCILLOSCOPE	Displays wave shape vs: time on its monitor from signals received from inlet pressure of any transducer or from drum or pick off from the 9-196-01 UUT.
E33	SCOPE INPUT Jack	Receives input signals from E36.
E34	REMOTE FLOW INPUT Jack	Receives external 0-5VDC signal for external control of flow.
E35	MODE SELECT Rotary Switch	Selects the system test operating mode desired: STANDBY, AUTO, MANUAL or REMOTE.
E36	SCOPE INPUT SELECT Rotary Switch	Selects the scope input signal desired: INLET PRES-SURE, DRUM or PICKOFF.
E37	MASTER FLOW Indicator	Displays the true fuel flow in pounds per hour (PPH), after compensating for temperature and specific gravity. Communicates results with computer E700.
E38	OPERATOR TERMINAL #1	Interfaces with computer E700 and also acts as its keyboard. Displays the following: UUT flow angle in degrees, UUT flow in PPH, Actual flow in PPH, Differ-ential flow between UUT and actual flow in PPH, UUT power in VA, UUT Setup Messages Response time during Response Test in seconds, Displays

Table 5-1. Electrical Controls and Instruments - Continued

ITEM NO	DESCRIPTION	FUNCTION
E38	OPERATOR TERMINAL #2 - (Continued)	instructions during UUT or System Programming.
E39	START FLOW Switch	Directs fuel flow through the UUT.
E40	STOP FLOW/BYPASS Switch	Used while performing response tests only. It bypasses fuel flow around the UUT.
E41	PROGRAM/RUN Keyswitch	Normally placed in RUN position for testing. When turned to PROGRAM position, it allows UUT and System variables to be programmed.
E42	OPERATOR TERMINAL #2	Interfaces with computer E700 and also acts as its keyboard. Displays and controls the following: Flow - from 0 to 120,000 PPH, Back pressure - from 0 to 25 PSI, Data entry during UUT or System Programming, and System Messages.
E50	EMERGENCY STOP Pushbutton (Above sink area)	Enables shutdown of control power to test stand.
E505	PURGE TIMING Timer Relay (In Start/Stop Box)	Adjustable between 1-30 minutes for timeout delay.
E509	UUT TEST Connector (Sink Area)	Using specified UUT adapter cable, links UUT to electrical control cabinet controls and instruments.
E516	MAIN PUMP MOTOR Starter (Pump Section)	Automatically starts and stops the main pump motor E601. Press the RESET pushbutton on the starter cover in case of motor stoppage due to overload.

Table 5-1. Electrical Controls and Instruments - Continued

ITEM NO	DESCRIPTION	FUNCTION
E520	RESIDUE TRANSFER PUMP TIMER RELAY (Electrical Cabinet)	Adjustable between 0.6-60 seconds for delay on release after level switch E507 opens.
E554	SINK LIGHT Switch (Sink Area)	Turning the switch to ON position, turns on the sink light E554. Turning to OFF position switches off the sink light.

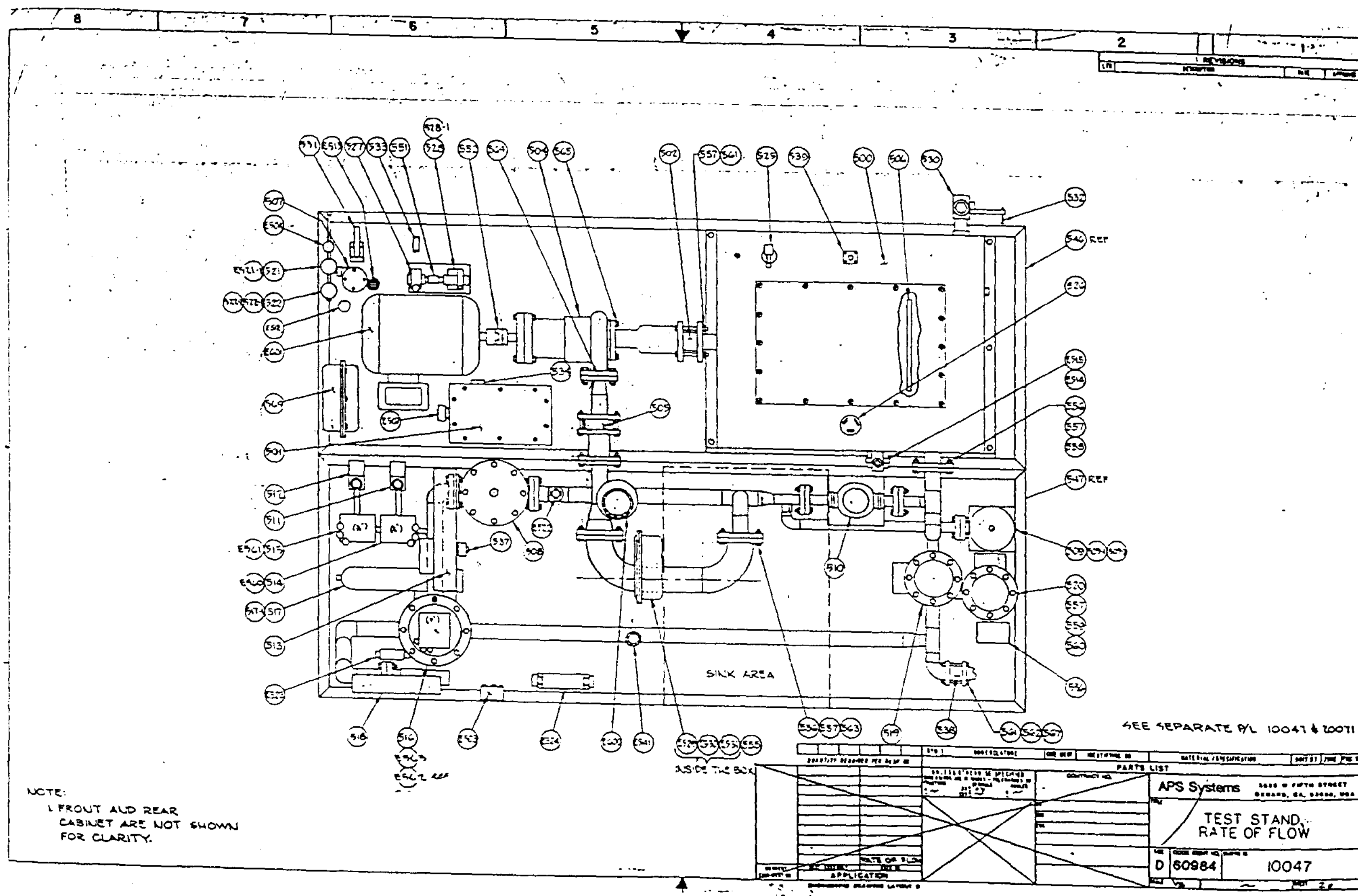


Figure 5-2. Test Stand Mechanical Controls and Instruments

Table 5-2. Mechanical Controls and Instruments

ITEM NO.	DESCRIPTION	FUNCTION
502	Shut-Off Valve	Enables outlet of fuel from reservoir 500 to pump 504.
508	Main Filter	10 microns fuel filter before air operated shut-off valves 511, 512 and 513.
509	Accumulator	Pulsation Dampening. Must be precharged to 25 PSIG.
521	Air Filter Regulator With Gage	Controls shop air outlet pressure by turning adjusting knob clockwise to increase or counterclockwise to decrease air pressure. Sets outlet pressure at 90 PSIG. Drain liquid from regulator frequently and clean or replace filter element when dirty.
522	Coalescing Filter	Drain liquid from filter frequently. When its service life indicator rotating ball shows half red/green position, replace filter element.
523	Fuel Shut-Off Valve	Open only when filling fuel in reservoir 500 by pressure.
529	Air Pressure Regulator With Gage (In electrical control cabinet)	Controls shop air outlet pressure by turning adjusting knob clockwise to increase or counterclockwise to decrease air pressure. Sets purge air pressure at 0.15" of water.
530	Fuel Drain Shut-Off Valve	Used to drain fuel reservoir 500.
531	Water Shut-Off Valve	When opened, allows facility water to enter heat exchanger 506.
534	Waste Reservoir Sight Glass	Used to view level of fuel in waste reservoir 501.

Table 5-2. Mechanical Controls and Instruments - Continued

ITEM NO.	DESCRIPTION	FUNCTION
535	Shut-Off Valves (See Figure 4-1)	Two of them are opened to take inlet, differential and back pressure readings. Another one is opened to take fuel samples or for purging fuel system.
539	Fuel Level Gage	Displays fuel level in reservoir 500.
542	Four Way Air Valve (See Figure 4-1)	Used to clamp or remove UUT from clamping cylinder 541.
543	Air Pressure Regulator With Gage (In electrical control cabinet)	Controls shop air outlet pressure by turning adjusting knob clockwise to increase or counterclockwise to decrease air pressure. Sets I/P transducers air outlet pressure at 20 PSIG.

CHAPTER VI

OPERATION

6-1 GENERAL

The following paragraphs describe safety features, test modes, startup and shutdown procedures, emergency shutdown, operator terminal features and programming of the test stand. When reading the instructions and procedures, refer to the hydraulic and electrical schematics (Figures 4-1 and 4-2) and Figures 5-1 and 5-2.

6-2 SAFETY FEATURES

The test stand is equipped with safety features that protect the operator, test stand and the Unit Under test (UUT). These safety features, described in the following paragraphs, will function to either shutdown the test stand or give warning about equipment malfunction due to the unsafe condition.

6-2-1 Annunciators.

Most abnormal conditions are monitored from signals detected by pressure, temperature, flow and level sensing switches. These are displayed by illumination of the red System Fault light E21, sounding of the audible alarm E22 and display of messages on operator terminals E38 and E42, all located on the front electrical control cabinet panels.

a. Facility Air Pressure Low. If the facility supplied air pressure drops below 70 PSIG, the contacts of facility air pressure switch E513 will open. This will illuminate the red light E21, sound the alarm E22, display fault messages on terminal #2, E42 and disable pump operation.

b. Water Supply Pressure Low. If the water pressure from the facility drops below 40 PSIG, the contacts of facility water pressure switch E512 will open. This will illuminate the red light E21, sound the alarm E22, display fault messages on terminal #2, E42 and disable pump operation.

c. Fuel Reservoir Temperature High. If the temperature of fuel in reservoir 500 rises to 90°F, the contacts of fuel reservoir temperature switch E514 will open. This will illuminate the red light E21, sound the alarm E22, display fault messages on terminal #2, E42 and shutdown power to the test stand.

d. Fuel Reservoir Level Low. When the fuel level in the reservoir 500 falls below 1/2 full level, the contacts of main fuel reservoir level switch E515 will open. This will illuminate the red light E21, sound the alarm E22, display fault messages on terminal #2, E42 and shutdown power to the test stand.

e. Fuel To UUT Temperature High. If the temperature of fuel flowing to the UUT as indicated by thermocouple E522 rises to 90°F and cannot be reduced by temperature controller E15 signals, the red light E21 illuminates, alarm E22 sounds, fault messages are displayed on terminal #2, E42 and pump operation disabled.

f. Vent Flow Stops. If the exhaust flow from both the test stand sections and the sink area stops, the contacts of vent fan flow switch E541 will open. This will turn on the red light E21, sound the alarm E22 and display fault messages on terminal #2, E42. The test being performed can be completed with the vent fan motor E600 not working.

6-2-2 Emergency Stop Switches.

Two palm operated switches are located in easily reached positions to permit immediate test stand shutdown in event of an emergency. One switch E23 is mounted on the front panel of electrical control cabinet and the other E50 is located above the test chamber sink area. A third fusible link sensor switch E548-4 automatically shuts down the test stand in case of fire or overheated test stand or test chamber area. The switches are wired in series in the test stand power control circuit and are normally closed. When any of the emergency switches E23 or E50 is pressed or the sensor switch E548-4 opens, it removes 115 VAC power from all control circuits in the test stand.

6-2-3 Purge Pressure Failure

Pressing the SYSTEM START pushbutton E2 starts the electrical control cabinet purge time out. At any time if sufficient purge pressure is not maintained, electrical control power will be removed from the stand.

6-2-4 Pump Motor Overload Protection

Main pump motor E601 is protected by thermal overload E517 built into the main pump motor starter E516 circuit. An overheated motor can be restarted after the motor has cooled by resetting the thermal switch on motor starter E516 cover.

6-3 SYSTEM TEST MODES OF OPERATION

System test operational modes are selected by using the front panel MODE SELECT rotary switch E35. The modes selected are: STANDBY, AUTOMATIC, MANUAL or REMOTE. The modes are described in the following paragraphs.

6-3-1 Standby Mode

In the STANDBY mode, the system could be started, programmed or is in the alarm processing mode. During installation and removal of UUT, it removes all UUT power and flow.

1. It provides System Status and alarm messages on both terminals E38 and E42.

2. It allows selection of UUT and displays a summary of all setup conditions for test stand equipment which include:

   a) UUT Test Order Number.
   b) UUT Adapter Cable Number.
   c) UUT Power Voltage, Current and Phase settings.
   d) Fuel Temperature Set Point.

3. It forces flow to zero.

4. It allows modification or entry of new UUT or System Data. Security of the data is ensured by PROGRAM/RUN keyswitch E41 and the PASSWORD protection.

5. It also performs simple self test on initial power up of test stand.

6-3-2 Automatic Mode

AUTOMATIC mode is the normal mode during operation and testing of UUT.

1. It allows operator control of flow by keyboard entry of desired value, or by incremental increase or decrease of flow from present value.

2. It automatically limits maximum flow to preprogrammed value for particular UUT.

3. It automatically selects appropriate signal conditioning and scale factors for displaying accurate UUT flows.

4. It automatically selects proper flow rate controller 514 for 40 to 590 Pounds Per Hour (PPH) flow, or 515 for 400 to 7,700 PPH flow or 516 for 5,800 to 115,000 PPH flow according to operator requested flow value. It also automatically sequences flows through zero when changing to a higher or lower flow rate controller.

5. It automatically determines flow thresholds and response intervals during transducer response tests.

6. It displays the following:

a) UUT Flow in PPH.
b) UUT Flow Angle in degrees.
c) Actual flow in PPH.
d) Difference in flow between indicated and actual in PPH.
e) UUT Power in Voltage, Current and calculated VA.
f) UUT Response Time in seconds.
g) UUT Inlet Pressure and Back Pressure in PSIG and Differential Pressure in PSID.
h) Test Order Number for UUT.
i) Fuel Temperature in ^{o}F.
j) Waveform of UUT Inlet pressure for all transducers, or waveform of Drum or Pickoff for Type 9-196-01 transducer.
k) Amplitude of induced Vibration in Mills Peak.
l) It sounds the audible alarm E22 and illuminates SYSTEM FAULT light E21 during alarm conditions.

6-3-3 Manual Mode

In the MANUAL mode it performs identical to the AUTOMATIC mode, except UUT back pressure is controlled by the operator.

6-3-4 Remote Mode.

In the REMOTE mode, it allows flow value to be controlled remotely using a 0 to +5VDC external signal.

<u>CAUTION</u>

The REMOTE mode can cause damage to the UUT. Do not attempt to use this mode without approval and supervision of the cognizant Test Engineer.

This mode is for engineering test only and does not provide any automatic control or limiting of UUT flow or back pressure.

6-4 STARTUP PROCEDURE

After the test stand has been installed and set up according to paragraphs 3-5 through 3-7, the following should be checked:

1. Ensure the fuel used for testing is MIL-C-7024B Type II fluid.

2. Check amount of fuel in reservoir 500. Fill it to between 3/4 and full as shown on level gage 539 by either of the following methods:

a) Pressure fill through shut-off valve 523 and low pressure filter 524.

b) Manually fill through safety fill cap 525 on top of reservoir 500.

CAUTION

Before proceeding further, be sure that facility services are connected in accordance with Chapter III. Be sure there are no leaks at service connections.

NOTE

When measuring voltages at motor, the indication across any two of the three lines will be 480 volts and 115 volts at electrical control cabinet.

3. Measure and verify facility power to the motor E601 for 480VAC and 115VAC to the electrical control cabinet.

4. Accumulator 509 must be precharged with Nitrogen gas at 25 PSIG.

5. Ensure facility air is available above 90 PSIG.

6. Ensure facility water is available at 40 PSIG minimum pressure and 60°F maximum temperature.

7. Open shut-off valve 531 to allow water to flow through to heat exchanger 506.

8. Open fuel shut-off valve 502 to allow fuel to flow from reservoir 500 to main pump 504.

9. Check for leaks in air, water and fuel systems. Repair as necessary.

10. Turn MODE SELECT rotary switch E35 to STANDBY position.

11. Press SYSTEM START pushbutton E2. Red PURGE FAIL light E1 should be off.

12. After approximately 2 minutes (purgeout period), power will automatically be available to front panel of electrical control cabinet.

13. Turn the power ON switches to LOW FREQUENCY power supply E8, phase angle indicator E31, TRANSIENT MONITOR oscilloscope E32 and MASTER FLOW indicator E37.

14. On operator terminal #2, E42 select Test Code for UUT.

15. On operator terminal #1, E38 sequence through UUT setup messages and adjust setup accordingly:

a) Connect UUT to UUT TEST connector E509 using specified UUT adapter cable.

b) Select desired frequency and phase at LOW FREQUENCY power supply E8.

c) Adjust to desired voltage at power supply E8.

d) Install UUT in transmitter clamping cylinder 541 using adapters and secure using Four-Way air valve 542.

e) Adjust FUEL TEMPERATURE controller E15 set point to temperature desired (usually 80°F). Use UP and DOWN arrows to set desired temperature. The press "ENTER".

16. Turn MODE SELECT rotary switch E35 to AUTOMATIC or MANUAL position.

17. Press PUMP/UUT POWER ON pushbutton E17. Amber PUMP/UUT ON light E18 should illuminate.

18. Press START FLOW switch E39 to start fuel flow. Run for approximately 30 minutes in order to allow test stand to warm up.

19. Perform UUT test in accordance with appropriate Test Order:

a) On operator terminal #2, E42 enter flow value.

b) On operator terminal #1, E38 read and record angle, actual flow, UUT flow and differential flow readings.

c) Repeat for next flow value in Test Order until completed.

d) Perform Response test, if required in Test Order as follows:

1) On operator terminal #2, E42 enter flow value.

2) Press STOP FLOW/BYPASS switch E40 to bypass fuel to UUT.

3) Wait for system to settle using terminal #1, E38 as an interval timer.

4) Press START FLOW switch E39 to start fuel flow through UUT.

5) Wait for flow to reach the required value.

6) On terminal E1, E38 read and record the response time.

e) On operator terminal #1, E38 measure UUT power.

20. On operator terminal #2, E42 enter zero flow value.

6-5 SHUTDOWN PROCEDURE

To shutdown the test stand, perform the following:

CAUTION

Do not use emergency stop switches for normal shutdown. Sudden shutdown of the Test Stand under high-pressure and RPM subjects components to undue stress and wear.

1. Enter zero flow on Terminal #2, E42 and press STOP FLOW/BYPASS switch E40 to stop flow.

2. Press PUMP/UUT POWER OFF pushbutton E16 to turn pump off. Amber PUMP/UUT ON light E18 should extinguish.

3. Turn MODE SELECT rotary switch E35 to STANDBY position.

4. Disconnect and remove UUT adapter cable connecting UUT to UUT TEST connector E509.

5. Remove UUT from clamping cylinder 541 using Four-Way air valve 542.

6. Press SYSTEM STOP pushbutton E3 to shut off power to electrical controls and instrumentation.

7. Close fuel shut-off valve 502 to shut off fuel flow from reservoir 500.

8. Close shut-off valve 531 to shut off water supply to heat exchanger 506.

9. Close facility's air, water and electrical supplies to test stand.

6-6 EMERGENCY SHUTDOWN

If it is necessary to only stop the test in progress, that can be done by pressing PUMP/UUT POWER OFF pushbutton E16. If required to completely stop power to control and instrumentation circuit, that can be done by pressing either SYSTEM STOP pushbutton E3, or EMERGENCY STOP pushbuttons E23 or E50.

6-7 READING DATA FROM OPERATOR TERMINAL #1.

Operator terminal #1, E38 normally displays messages or operator selected data. Sequencing is accomplished by pressing a key repeatedly. The data display selected is indicated by light next to that key. Type of data is indicated in display with an abbreviated english language descriptor as follows:

KEY	FUNCTION
FLOW	Sequences through four different displays of flow: 1) UUT Flow Angle - in degrees. 2) UUT Flow - in PPH. 3) Actual Flow - in PPH. 4) Flow Difference between actual and indicated in PPH.
POWER	Displays UUT power calculated as Voltage times Current. Sequences through two displays.
SETUP	1) UUT Code Number (ranges from 10 to 50). 2) Test Order Number. 3) UUT Adapter Cable Number. 4) UUT Voltage - 400 Hz. 5) UUT Voltage Low Frequency. 6) UUT Frequency. 7) UUT Phase. 8) Vibration Level in mills. 9) Fuel Temperature Set point.

6-8 CONTROLLING RESPONSE TEST FROM OPERATOR TERMINAL #1.

Operator terminal #1, E38 controls the automatic response test procedure. The selected testing mode is indicated by light next to that key, and modes selected are:

KEY	FUNCTION
NORMAL	Normal flow operation - used for all testing except Response Test.
FREEZE	Temporarily freezes all control functions to let flow settle before response test.
RESPONSE	Enables Response Mode Test with time interval displayed on terminal #1, E38.

6-9 DISPLAYING MESSAGES FROM OPERATOR TERMINAL #2.

When faults occur within the system the system fault light E21 and audible alarm E22 will be activated. Dangerous faults will automatically shut off the main pump 504. When the alarm sounds, return to STANDBY using rotary switch E35 and press "MESSAGE" key on operator terminal #2, E42. This turns off the audible alarm E22 and displays the fault on operator terminal #2, E42. See the troubleshooting chapter IX for remedy. After the fault is rectified, the fault message should be cleared by pressing "CLEAR" on operator terminal #2, E42.

6-10 ENTERING DATA FROM OPERATOR TERMINAL #2

Operator terminal #2, E42 allows keypad input of control or programming data.

6-10-1 Entering A Flow Or Back pressure Value.

1. Input the desired value from left to right using the 0-9 keys and "." if appropriate. Begin with the most significant number. End with the least significant number. When the value is correct, press "ENTER" to initiate this new value.

OR

2. Press the UP or DOWN arrow key to increment or decrement the present Flow value.

KEY	FUNCTION
"0"-"9"	Enters the number in the next character position.
"."	Places a decimal point in the next character position.
"DELETE"	Deletes the previous position character.
"SPACE"	Places a space in the display. NOTE: Do not use this for number entries.
"CLEAR"	Clears all characters in display.
"UP arrow"	Increments present value by approximately 2% of the maximum allowable value (of flow or back pressure). Hold down key to repeat increment.
"DN arrow"	Decrements present value by approximately 2% of the maximum allowable value (of flow or back pressure). Hold down key to repeat decrement.

All other keys are ignored.

6-10-2 Entering A Message Or Alpha Numeric Value (During Programming) -

Input the desired message or value into the display from left to right using any of the keys listed below.

When the display is correct, press "ENTER" to complete the entry.

NOTE: To enter an alphabetical letter in a position:

1. First enter a space, dash or number.

2. Then modify this character using the UP and DOWN keys.

KEY	FUNCTION
"0"-"9"	Enters the number in the next character position.
"."	Place a decimal point in the next character position.
"DELETE"	Deletes the previous position character.
"SPACE"	Places a space in the next character position. NOTE: Do not insert spaces in numbers.
"-"	Places a dash in the next character position. NOTE: Do not insert dashes in numbers.
"CLEAR"	Clears all characters in display.
"UP arrow"	Inserts an alphabetical letter (A-Z) into the LAST character position. Hold down key to increment from A to Z. NOTE: Do not use letters when entering a number.
DN arrow"	Inserts an alphabetical letter (A-Z) into the LAST character position. Hold down key to decrement from Z to A. NOTE: Do not use letters when entering a number.

All other keys are ignored.

6-11 PROGRAMMING UUT DATA AND SYSTEM VARIABLES

The Rate of Flow test stand contains an embedded microprocessor which automatically configures valves and equipment within the stand to provide the correct flows and signal processing for the UUT selected. Programming of new configuration variables, or modifying of existing variables, is accomplished using the two front panel Operator Terminals #1 and #2, E38 and E42 and the PROGRAM/RUN Keyswitch E41.

CAUTION

To prevent damage to equipment this procedure should only be performed by a qualified Test Engineer. Any changes to the existing program must be approved and documented for configuration control.

6-11-1 Programming Set Up

Programming can only be performed in the STANDBY mode after power is made available to electrical cabinet. Select PROGRAM using keyswitch E41. Then select the Programming GROUP using the Operator Terminal #2, E42 "MESSAGE" key. The programming groups are:

1. UUT Data - Modify or enter new configuration and signal processing variables for UUTs.

2. SYSTEM Data - Modify test stand variables associated with closed loop control of flow and pressure, independent of UUT.

3. CALIBRATION - Performs an automatic recalibration of flow and back pressure. This should be performed when master flow values do not match the operator requested values. Takes approximately 30 minutes. Requires a calibration jig. (See paragraph 10-3).

4. SAVE Data - Permanently stores new data or calibration values generated during the programming mode. This step must be performed before returning to normal test operation.

6-11-2 Programming UUT Data

1. Select UUT DATA using the "GROUP" key.

2. Enter UUT code number to be modified or a new UUT code number.

NOTE

New UUT codes must be programmed in numerical sequence starting with one number higher than the latest programmed code. It is the PROGRAMMERS responsibility to assure the correct numbering. Consult the configuration documentation before attempting to do this.

3. Using the "ITEM" key, select the ITEM to be modified

by sequencing through the following list. Stop at the desired item. (See Table 6-1).

4. Enter new value or modify the existing value. Then press "ENTER" to temporarily store the change.

5. The UUT Conversion Table consists of up to 24 pairs of numbers which are used to automatically convert UUT angle to flow. Use the UP arrow and DOWN arrow to sequence through the numbers (up to 48 total).

NOTE

For each DEGREE number entered, you must follow with an equivalent Flow value - in PPH. These values are defined in the UUT Test Order. All DEGREE numbers in this table must be entered in ascending value starting with 0 degrees. After all Test Order values have been entered, the last entry must be 0 degrees and 0 PPH to indicate the end of valid data. It is the PROGRAMMERS responsibility to ensure this.

6. When all ITEMS for a particular UUT Code are properly entered, sequence to the UUT CODE and enter another CODE if desired and repeat all instructions for changing the ITEMS within this code.

7. When all UUT CODEs are successfully entered or modified, advance the GROUP to "SAVE DATA" and press "1" when requested to permanently save the changes. If mistakes are made in entering new data, you may ignore all changes by returning to normal test operation without SAVING the DATA.

NOTE

You MUST SAVE DATA to permanently STORE your programming changes.

8. Select RUN on the PROGRAM/RUN keyswitch E41 to return to normal testing operation.

Table 6-1. UUT Data Items

ITEM	TYPE*	MAX # CHARACTERS	COMMENTS
UUT Code	Integer	2	10 through 99 must must be in sequence
TEST ORDER Number	ASCII	16	Any characters
ADAPTER CABLE	ASCII	11	Any characters
UUT power VOLTS 400HZ	ASCII	4	i.e. 115 etc.
UUT power VOLTS low frequency	ASCII	4	i.e. 15
UUT power low HZ	ASCII	5	i.e. 400 etc.
UUT power low PHASE	ASCII	2	i.e. 1, 2 or 3
UUT Vibration Level	ASCII	5	i.e. 2.0
TEMPERATURE-degrees F	ASCII	5	i.e. 80
MAX FLOW	Integer	6	50 to 120,000 PPH
UUT MAX FLOW	Integer	6	i.e. 80,000
XDUCER TYPE	Integer	1	0, 1, 2, or 3 where: 0 = 3 Phase synchro 1 = Second Harmonic 2 = Magnetic pickup 3 = 3 Phase synchro with zero flow at 280°
RESPONSE WAIT TIME	ASCII	3	Recommended settling time
UUT Conversion Table (Enter up to 24 points)			
Degrees	Number	6	Fractions ok
PPH	Integer	6	50 to 120,000 PPH

6-11-3 Programming System Data

NOTE

THIS PROCEDURE SHOULD ONLY BE PERFORMED BY A CERTIFIED TEST ENGINEER. CONSULT FACTORY FOR ASSISTANCE.

1. Select SYSTEM DATA using the key.

2. Enter PASSWORD to gain access to this feature. The Password is provided in software maintenance manual and should only be given to authorized personnel.

3. Using the "ITEM" key, select the ITEM to be modified.

4. Enter new data by sequencing through the following list. Stop at the desired item. (See Table 6-2).

5. When all SYSTEM DATA ITEMS are successfully entered or modified, advance the GROUP to "SAVE DATA" and press "1" when requested to permanently save the changes. If mistakes are made in entering new data, you may ignore all changes by returning to normal test operation.

NOTE

You MUST SAVE DATA to permanently STORE your programming changes.

6. Select RUN on the PROGRAM/RUN keyswitch E41 to return to normal testing operation.

Table 6-2. System Data Items

ITEM	INITIAL VALUE
Back pressure Loop Gain	0.3
Back pressure Integral	0.2
Back pressure Differential	0.5
Back pressure Update Time	0.5
Back pressure Display Gain	20
Back pressure Calculation Exponent	1.65
Back pressure Minimum Pressure	1.6
Value 1 Maximum	N/A
Value 1 Minimum	N/A
Value 2 Maximum	N/A
Value 2 Minimum	N/A
Value 3 Maximum	N/A
Value 3 Minimum	N/A
Harmonic Conditioner Gain	30.297
Harmonic Conditioner Offset	0
9-196-01 UUT Drum Gain	360
9-196-01 UUT Drum Offset	0
External Input Flow Gain	0.197

6-11-4 Calibration of Automatic Flow and Back Pressure. For calibration of automatic flow and back pressure see paragraph 10-3.

6-12 UUT ASSIGNMENTS

Each UUT pin to be tested is assigned a unique testing CODE Number and Adapter Cable Number. The CODE number is keyed in by the operator to automatically configure the measurement and display equipment. The adapter cable electrically mates the various UUTs to the single UUT interface connector E509 on the test stand. The CODE and Adapter Cable assignments are listed in Table 6-3.

Table 6-3. UUT Assignments

APS ASSIGNED CODE NUMBER	APS ADAPTER CABLE NUMBER	UUT PART NUMBER	APS FLUID ADAPTER NUMBERS
10	N/A	N/A	N/A
11	70018(E900)	8TJ50GBM5	40999-03
12 12	70019(E901) 70019(E901)	8TJ62GBA3 8TJ62GBK3	40999-03 40999-11
13 13	70019(E901) 70019(E901)	8TJ62GBC3 8TJ62GCA3	40999-07 40999-05
14	70019(E901)	8TJ62GBZ3	40999-07
15	70020(E902)	8TJ64GDM2	41004-01
16	70036(E903)	8TJ85GHM2	40999-13 & 40999-03
17	70037(E904)	9115-16A1A	41004-01
18 18 18 18	70037(E904) 70037(E904) 70037(E904) 70037(E904)	9115-16B1A 9115-16C1A 9115-16C4A 9115-16D1	41004-01 41004-01 41004-01 41004-01
19	70044(E905)	9117-16A1	41004-01
20	70037(E904)	9121-21A1	41004-01 & 40889
21	70045(E906)	150-005-003	40999-01 & 40889
22	70046(E907)	9-196-01	40999-13
23 23	70018(E900) 70018(E900)	8TJ50GAS4 8TJ50GAS5	40999-03 40999-03
24	Reserved for test and setup.		

NOTE: Use appropriate size gaskets listed in Table 2-1.

CHAPTER VII

COMPUTER DOCUMENTATION

7-1 GENERAL

The software and hardware documentation for the Fuel Rate of Flow Test Stand has been placed in the appendix of this manual. It covers the Software Description in Section 1, Software Flow Charts in Section 2, Software Listing in Section 3, MID Reference Manual in Section 4 and Computer Hardware Configuration which is required for the software to operate properly in Section 5.

CHAPTER VIII

PERIODIC INSPECTION AND MAINTENANCE

8-1 GENERAL

The Fuel Rate of Flow Test Stand should be inspected and serviced on a regular basis. This chapter contains the recommended instructions and information for maintenance of the test stand. Routine performance of the inspection and maintenance procedures will keep the stand operating according to specification and increase its service life. For specific procedures pertaining to the maintenance of components, refer to the commercial literature provided.

8-2 ROUTINE MAINTENANCE

Prior to test startup, the test stand should be inspected to assure proper test stand performance. Service and clean as required by the following checks:

a. Check oil level in the reservoir.

b. Inspect all drains. They must be free of debris. Remove all debris from the test stand base.

c. Inspect for cleanliness in the sink area.

d. Inspect the backside of the sink area and the bottom of test stand for any trace of fluid leakage.

Observe the test stand at all times for pneumatic and fuel leaks, burned out indicators, loose equipment, defective hoses and similar minor faults. Correct these as soon as they are detected before they become major malfunctions.

8-3 PERIODIC MAINTENANCE

The test stand should be serviced on a regular basis. Table 8-1 lists the components to be checked, what to look for, the frequency of service and the instructions for performing maintenance on the test stand.

WARNING

To prevent serious injury or death to personnel, shut off all services to the test stand before performing any maintenance or servicing to the test stand.

Table 8-1. Periodic Maintenance Chart

Inspection Item	For	Frequency	Remarks
Fluid reservoir 500 liquid level	Check fluid level on level gage 539.	Before operation	Fill when below 3/4 full mark.
Regulator 521	Check gage on regulator 521.	During operation	Regulator should be set at 90 psig.
Regulator 529	Check gage on regulator 529.	During operation	Regulator should be set to maintain 0.15" WC purge pressure.
Regulator 543	Check gage on regulator 543.	During operation	Regulator should be set at 20 psig.
Indicator lights	Observe lights when their circuits are activated.	During operation	If defective, replace bulbs.
Moisture in air system	Open drain valves of air filters 521 & 522.	Weekly	Drain moisture and close valves.
Internal leaks	Open maintenance doors and visually check entire stand for leaks.	During operation	Tighten or replace leaky joints.
Valve packing	Check valve bonnets for leakage.	Monthly	Tighten packing nut or replace packing.
Filters 508, 521, 522 and 524	Observe clogging indicator.	Monthly	Replace filter elements, if dirty.
Pressure relief valves 510 & 553	Connect pressure source as required.	Annually	If necessary, reset valve using adjustment on valve body. Relief valve

Table 8-1. **Periodic Maintenance Chart-Continued**

Inspection Item	For	Frequency	Remarks
Pressure relief valves 510 & 553 - (Continued)			settings should be: 510 @ 85 psig and 553 @ 0.15 inch of water pressure.
Vent motor E600 and Pump motor E601	Lubrication per MIL-L-7711	800 hours	Pressure grease fitting on each end of motor.

8-4 CHECKOUT ITEMS

It is recommended that the following functions and settings be checked out at least every six months. At such time, calibrate, adjust and repair as necessary.

a. Startup Procedure in accordance with paragraph 6-4.

b. Settings of Pressure Controls listed in Table 10-2.

c. Test air pressure and fuel oil pressure, as well as other test parameters, by operating the test stand with a transmitter of known performance.

d. Pressure transducers and flow transducers.

e. Any Pressure Gage suspected of mis-calibration.

f. Repeat steps b through e if necessary to achieve the proper test stand operation specification.

g. If maintenance or a simple repair is required, perform the shutdown procedure given in paragraph 6-5 before beginning maintenance or repair.

h. Recalibrate flow and back pressure controls per paragraph 10-3.

8-5 FACTORY REQUIRED MAINTENANCE SCHEDULE

The Factory Required Maintenance Schedule (Figure 8-1) lists components requiring service and tasks to perform. Look at the top row. Across, are written numbers 1 through 12; following sheets reach 36. Each number stands for a month.

Example of use: At the end of the first month, look below the number 1 to the first blank box. In this case, the first box is in the same row with this note:

P	**Air Filters** **Clean;Replace Element**	**1**

Perform the task. Initial the box under number 1. NOTE: Leading Letters: P: Pneumatic System; F: Fuel System; H: Water System; E: Electrical System.

Continue looking down the list. Skip shaded areas. At each blank box, perform the task listed across from it. Perform all tasks to be completed for that month. At the end of month 2, look below the number 2 and proceed in like manner. For frequency of maintenance, 1 month was deemed equivalent to 160 hours of service. Under more intense service, proportionately increase the frequency of maintenance.

FIG. 8-1 FACTORY REQUIRED MAINTENANCE SCHEDULE

		1	2	3	4	5	6	7	8	9	10	11	12
P	**Air and Coalescing Filters** Clean;Replace Element												
P	**Pressure Regulators** Check Settings												
P	**Pressure Switch** Check Setting	//////	//////		//////	//////		//////	//////		//////	//////	
H	**Heat Exchanger** Check for Cracks and Pinholes	//////	//////		//////	//////		//////	//////		//////	//////	
H	**Pressure Switch** Check Setting	//////	//////		//////	//////		//////	//////		//////	//////	
F	**Filters** Blowdown. Cln/Replace Element												
F	**Pressure Gages** Calibrate	//////	//////		//////	//////		//////	//////		//////	//////	
F	**Pressure Regulators** Check Settings												
E	**Flow Transducers** Calibrate	//////	//////		//////	//////		//////	//////		//////	//////	
E	**Pressure Switches** Check Settings; Adj.As Req.	//////	//////	//////	//////	//////		//////	//////	//////	//////	//////	
E	**Pressure Transducers** Calibrate	//////	//////		//////	//////		//////	//////		//////	//////	
E	**Temperature Switch** Check Setting; Adj.As Req.	//////	//////		//////	//////		//////	//////		//////	//////	
E	**Level Switches** Check Settings; Adj.As Req.	//////	//////		//////	//////		//////	//////		//////	//////	
E	**Limit Switches** Check Settings; Adj.As Req.	//////	//////		//////	//////		//////	//////		//////	//////	

FIG. 8-1 FACTORY REQUIRED MAINTENANCE SCHEDULE

		13	14	15	16	17	18	19	20	21	22	23	24
P	**Air and Coalescing Filters** Clean;Replace Element												
P	**Pressure Regulators** Check Settings												
P	**Pressure Switch** Check Setting												
H	**Heat Exchanger** Check for Cracks and Pinholes												
H	**Pressure Switch** Check Setting												
F	**Filters** Blowdown. Cln/Replace Element												
F	**Pressure Gages** Calibrate												
F	**Pressure Regulators** Check Settings												
E	**Flow Transducers** Calibrate												
E	**Pressure Switches** Check Settings; Adj.As Req.												
E	**Pressure Transducers** Calibrate												
E	**Temperature Switch** Check Setting; Adj.As Req.												
E	**Level Switches** Check Settings; Adj.As Req.												
E	**Limit Switches** Check Settings; Adj.As Req.												

FIG. 8-1 FACTORY REQUIRED MAINTENANCE SCHEDULE

		25	26	27	28	29	30	31	32	33	34	35	36
P	**Air and Coalescing Filters** Clean;Replace Element												
P	**Pressure Regulators** Check Settings												
P	**Pressure Switch** Check Setting												
H	**Heat Exchanger** Check for Cracks and Pinholes												
H	**Pressure Switch** Check Setting												
F	**Filters** Blowdown. Cln/Replace Element												
F	**Pressure Gages** Calibrate												
F	**Pressure Regulators** Check Settings												
E	**Flow Transducers** Calibrate												
E	**Pressure Switches** Check Settings; Adj.As Req.												
E	**Pressure Transducers** Calibrate												
E	**Temperature Switch** Check Setting; Adj.As Req.												
E	**Level Switches** Check Settings; Adj.As Req.												
E	**Limit Switches** Check Settings; Adj.As Req.												

CHAPTER IX

TROUBLESHOOTING

9-1 GENERAL

The information contained in this paragraph is intended to serve as a guide in locating and correcting troubles encountered in the operation of this test stand after the stand has been installed and checked out in its final location. The troubles listed are those that might be expected to occur, not as a result of improper operation of the test stand, but as a result of normal wear, adjustment drifting out of tolerance, failure of one or more components of the system, etc. It is assumed that the proper sequence of operation has been followed and that all special precautions have been observed.

The Troubleshooting Guide provided in Tables 9-1 through 9-5 is intended to be used with the test stand description in Chapter IV and the schematics and drawings in Chapters IV and XI. When used together, the information will aid in isolating the trouble to a component or group of related components. Once the problem has been isolated to a component(s), refer to the commercial literature provided for information on repair, calibration or replacement.

WARNING

Remove all power from the test stand before performing any electrical maintenance. Failure to do so could result in serious personal injury and/or damage to equipment.

CAUTION

Electrical equipment in the test stand is protected by fuses. An exception is the overload device on the motor starter which is reset manually. When an overload occurs, the contacts of the motor starter open and remain open until reset and the start pushbuttons are once again pressed. When a motor overload device has been tripped, the cause of the trouble should be located and corrected before the motor is placed back in operation.

Table 9-1. Troubleshooting Guide, Electrical

TROUBLE	PROBABLE CAUSE	REMEDY
1. Test stand control power cannot be turned on.	1. Loss of power on main supply line. 2. Fuses E500, E501 or E552 blown. 3. Open winding on SYSTEM START/VENT FAN relay E502. 4. SYSTEM STOP push-button E3 remains in open position.	1. Check power source and correct. 2. Replace fuses. 3. Replace relay. 4. Repair or replace defective pushbutton.
2. Stand power can be turned on, but de-energizes as soon as SYSTEM START push-button E2 is released.	1. Defective holding path for SYSTEM START/ VENT FAN relay E502.	1. Check holding contacts on relay. Repair/replace as necessary.
3. Stand power de-energizes automatically.	1. Loss of power on main power lines to stand. 2. Fuses E500, E501, E552 or E602 blown. 3. Motor E601 over-loaded.	1. Check power source and correct. 2. Replace fuses. 3. Check for overload condition or wiring fault. Press RESET pushbutton on motor starter E516.
4. Fuel pump POWER ON light E18 will not come on when push-button E17 is pressed.	1. Control power fuse E550 is blown. 2. Overload relay E517 at pump motor starter E516 tripped. 3. Facility air pressure switch E513 is open. Switch will open when air pressure is below 70 psig. Switch may be faulty or improperly set. 4. Water pressure switch E512 is open. Switch will open when water pressure is below 40 psig. Switch may be faulty or improperly set.	1. Replace fuse. 1. Check for excessive torque on pump shaft. Reset overload relay. 1. Determine cause and correct. 1. Determine cause and correct.

Table 9-1. Troubleshooting Guide, Electrical-Continued

TROUBLE	PROBABLE CAUSE	REMEDY
	5. Fuel reservoir temperature switch E514 is open. Switch will open when fuel temperature is above 90+/-2°F. Switch may be faulty or improperly set.	1. Determine cause and correct.
	6. Fuel reservoir level switch E515 is open. Switch will open when fuel level is below 1/2 full. Switch may be faulty or improperly set.	1. Determine cause and correct.
	7. Unit Under Test(UUT) fuel temperature controller E15 shows temperature above 90+/-2°F. Fuel is not cooled properly by heat exchanger 506.	1. Determine cause and correct.
5. No power to control chassis inspite of successful purgeout.	1. Fuse E532, E550 or E551 is blown.	1. Replace fuse.
6. One of the temperature, pressure, flow indicator, DC power supplies, etc. will not turn on.	1. Defective meter or indicator.	1. Determine cause and correct.

Table 9-2. Troubleshooting Guide, Electronic Controls

TROUBLE	PROBABLE CAUSE	REMEDY
MESSAGES:		
1. Low Water Pressure.	1. Pressure switch E512 open.	1. Apply cooling water. 2. Replace pressure switch E512.

Table 9-2. Troubleshooting Guide, Electronic Controls-Continued

TROUBLE	PROBABLE CAUSE	REMEDY
1. Low Water Pressure - (Continued)		3. Replace wiring. 4. Replace digital I/O rack E702 module #9. 5. Replace parallel I/O card E700-5.
2. Low Air Pressure.	1. Pressure switch E513 open.	1. Apply facility air. 2. Replace pressure switch E513. 3. Replace wiring. 4. Replace digital I/O rack E702 module #10. 5. Replace parallel I/O card E700-5.
3. High Fuel Temperature.	1. Temperature switch E514 open or temperature controller E15 sensing too hot due to insufficient cooling.	1. Increase water flow. 2. Reset temperature controller E15, measure temperature at UUT under flow. If okay: Replace temperature switch E514. 3. Replace digital I/O rack E702 module #11 and #14. Else: Replace thermocouple E522, or I/P transducer E533, or parallel I/O card E700-5.
4. Low Fuel Level.	1. Level switch E515 open.	1. Fill tank. 2. Replace level switch E515, or digital I/O rack E702 module #12 or parallel I/O card E700-5.
5. GPIB SEND ERROR.	1. Counter E546 communications lost.	1. Check cables. 2. Repair counter E546. 3. Replace interface card E700-4 or cables. 4. Repair phase angle indicator E31.
6. GPIB REC ERROR.	1. Phase angle indicator E31 or counter E546 communications error.	1. Check cables. 2. Repair phase angle indicator E31, or counter E546. 3. Replace interface card E700-4 or cables.

Table 9-2. Troubleshooting Guide, Electronic Controls-Continued

TROUBLE	PROBABLE CAUSE	REMEDY
7. GPIB TIME-OUT.	1. Phase angle indicator E31 or counter E546 communications error.	1. Check cables. 2. Repair phase angle indicator E31 or counter E546. 3. Replace interface card E700-4 or cables.
8. RS232 TIMEOUT.	1. Master flow indicator E37 or analog I/O rack E701 communications error.	1. Check cables. 2. Repair master flow indicator E37. 3. Verify/replace/repair all analog I/O rack E701 modules. 4. Replace interface card E700-3.
SYSTEM ERRORS (will disable all control)		
-1	1. File Open error or cannot open data file.	1. Reload software from 5-1/4" diskette. 2. Replace 20 MB hard drive or drive controller.
-2	1. File Read error or cannot read data file.	1. Reload software from 5-1/4" diskette. 2. Replace 20 MB hard drive or drive controller.
-3	1. File Close error or cannot close data file.	1. Reload software from 5-1/4" diskette. 2. Replace 20 MB hard drive or drive controller.
-4	1. File Write error or cannot write data file.	1. Reload software from 5-1/4" diskette. 2. Replace 20 MB hard drive or drive controller.
-5	1. Serial Open error - cannot initialize one of four serial COM ports.	1. Replace interface card E700-3.
-20	1. GPIB Initialization error.	1. Replace GPIB cables. 2. Remove phase angle indicator E31 and/or counter E546 cables - if error clears, repair E31 and/or E546. 3. Replace interface card E700-4.

Table 9-2. Troubleshooting Guide, Electronic Controls-Continued

TROUBLE	PROBABLE CAUSE	REMEDY
-21	1. GPIB Clear error.	1. Same as error 20.
-128 to 255	1. Action Pak communications error.	1. Verify-Replace all modules, subtract 128 from error displayed and consult manual for deciphering resulting error code.
Operator Terminal #1, E38 or #2, E42.	1. Dead.	1. Replace cables, terminal, or interface card E700-3.
Operator Terminals #1, E38 and #2, E42.	1. Both dead.	1. Replace fuse E700-7. 2. If 5VDC power missing - then replace power supply in E700. 3. Replace cables. 4. Replace interface card E700-3. 5. Turn off facility air - if system fault light E21 does not illuminate then replace computer system E700.
Flow Settings.	1. Inaccurate.	1. Perform auto calibration using program mode.
Back pressure Control.	1. None.	1. If back pressure indicator E20 reads okay, replace analog I/O rack E701 module #6. 2. Repair E20 & back pressure transducer E529.
Back pressure Control.	1. Unstable.	1. Perform auto calibration using program mode. 2. Reset system PID loop parameters - consult software description manual.
Type 1 UUT's.	1. Bad readings.	1. Replace analog I/O rack E701 module #9, or second harmonic converter E545.
Type 2 UUT's.	1. Inaccurate readings.	1. Adjust trigger levels of counter E546 channels until correct (usually set for 11:00 o'clock).

Table 9-3. Troubleshooting Guide, Pneumatic

TROUBLE	PROBABLE CAUSE	REMEDY
1. No air pressure is registered on filter/regulator gage 521.	1. Power to test stand has not been turned on.	1. Press SYSTEM START pushbutton E2.
	2. Air solenoid valve E504 defective.	1. Repair or replace solenoid valve.
	3. Filter 521 is clogged.	1. Clean filter. Replace element.
2. Inadequate air supply.	1. Regulator 521 is faulty or improperly set. Set regulator to 90 psig.	1. Adjust regulator; repair/replace as necessary.

Table 9-4. Troubleshooting Guide, Water System

TROUBLE	PROBABLE CAUSE	REMEDY
1. Improper cooling of fuel oil.	1. Shut-off valve 531 closed.	1. Open shut-off valve.
	2. No facility water supply to test stand.	1. Determine cause and correct.
	3. Temperature controller E15 or water regulating valve 507 are set incorrectly.	1. Adjust controller to 80°F; repair/replace as necessary.

Table 9-5. Troubleshooting Guide, Fuel System

TROUBLE	PROBABLE CAUSE	REMEDY
1. No fuel supply pressure, and POWER ON light E18 is off when pushbutton E17 is pressed.	1. Fault in electrical system.	1. Refer to Trouble 4 in Table 9-1.
	2. Low level of fuel in reservoir 500.	1. Replenish MIL-C-7024B, type II calibration fluid.
	3. Faulty level switch E515.	1. Adjust or replace as necessary.
	4. Shut-off valve 502 is closed.	1. Open valve.
	5. One-Way check valve 505 is stuck in closed position.	1. Replace check valve.
2. Unstable or improper fuel supply pressure.	1. Main filter 508 is dirty.	1. Clean filter and replace element, if necessary.
	2. Fault may be in pump 504.	1. Adjust, repair/replace as necessary.

Table 9-5. Troubleshooting Guide, Fuel System-Continued

TROUBLE	PROBABLE CAUSE	REMEDY
3. No fuel supply at UUT.	1. No air flow through solenoid valves E538, E539 or E540.	1. Determine cause and correct. 2. Computer E700 not sending signal to open the solenoid valves.
	2. None of the air operated shut-off valves 511, 512 or 513 are in the open position.	1. No air available to open air operated shut-off valves. See probable cause above.
	3. Diverter valve 517 stuck in the BYPASS position.	1. Computer E700 signalling air flow through bypass solenoid valve E536 and quick exhaust valve 537. 2. Bypass solenoid valve E536 malfunctioning. Repair/replace as necessary. 3. Faulty diverter valve 517. Repair/replace valve.
4. For response test, fuel supply flows through UUT only.	1. Diverter valve 517 stuck in flow position.	1. Bypass solenoid valve E536 not functioning. Check and replace, if necessary. 2. Quick exhaust valve 537 not functioning properly. Repair/replace as required. 3. Faulty diverter valve 517. Repair/replace valve.

CHAPTER X

CALIBRATION

10-1 GENERAL

This chapter contains the data for calibrating the instruments and gages. Before performing any adjustments on instruments or gages requiring calibration, consult the commercial literature furnished with the test stand.

10-2 CALIBRATION OF INSTRUMENTS AND GAGES

Pressure gages and other instruments will require periodic cleaning and calibration.

Calibration data for the gages and instruments are provided in Table 10-1. The recommended frequency of calibration is based on an average test stand usage of 100 hours per month. The actual calibration schedules should be established by the user. Preset Pressure Control Settings are provided in Table 10-2. Calibration data for flowmeters and pressure transducers is placed at the end of the chapter.

NOTE: The gages are supplied with a zero adjust which compensates the overall scale reading for reference shift; the zero adjust does not govern the span precision or the ability of the gage to repeat. Should the gage become defective as a result of use or abuse, then replace it with a properly calibrated gage of the same type and range.

To calibrate the gage, close the shutoff valve. Connect a line from the calibration port to the pressure source (a hydraulic dead weight tester is recommended). Apply pressure to the gage; verify the gage's reading within one percent (+/-1%) of the setting of the pressure source. To ensure complete calibration of the gage, repeat the verification at 10 points along the scale of the gage.

Table 10-1. Calibration Data

Item No.	Instrument	Range	Accuracy	Frequency
Test Equipment: Digital Multimeter HP3458A or equivalent.				
E4 E6	VOLTMETER	199.9 VTRMS	+/-0.5%	Annual
E5 E7	AMMETER	1.999 ATRMS	+/-0.5%	Annual
E8	POWER SUPPLY LOW FREQUENCY	VOLTAGE	+/-3%	Annual
E30	VOLTMETER	19.99 VAC	+/-0.6%	Annual
E547	POWER SUPPLY 400 Hz	115VAC	+/-2%	Annual
Test Equipment: Universal Counter HP5334B or equivalent.				
E8	POWER SUPPLY LOW FREQUENCY	Freq.	+/-1%	Annual
E547	POWER SUPPLY 400 Hz	400 Hz	+/-2 Hz	Annual
Test Equipment: Deadweight Tester AMETEK R-100 or equivalent.				
E13	INLET PRESSURE Indicator	0-75 PSIG	+/-0.5% of Full Scale	6 Months
E19	DIFFERENTIAL PRESSURE Indicator	+30 to -30 PSID	+/-0.5% of Full Scale	6 Months
E20	BACK PRESSURE Indicator	0-75 PSIG	+/-0.5% of Full scale	6 Months
E529	BACK PRESSURE Transducer	0-75 PSIG	+/-0.5% of Full Scale	6 Months
E530	DIFFERENTIAL PRESSURE Transducer	+30 to -30 PSID	+/-0.25% of Full Scale	6 Months
E531	INLET PRESSURE Transducer	0-75 PSIG	+/-0.5% of Full Scale	6 Months
Test Equipment: Send to Factory.				
E31	PHASE ANGLE Indicator	-180 to +180 Degrees	+/-0.001°	Annual

Table 10-1. Calibration Data-Continued

Item No.	Instrument	Range	Accuracy	Frequency
Test Equipment: Weight Rate Flow Calibration Equipment.				
E523	LOW FLOW Transducer	38-24,911 PPH	+/-0.1% of Reading	Annual
E524	HIGH FLOW Transducer	367-126,477 PPH	+/-0.1% of Reading	Annual
Test Equipment: Synthesizer HP8904A or equivalent.				
E546	UNIVERSAL TIMER Counter	10^{-6} sec		Annual
Test Equipment: Golden UUT.				
E546	UNIVERSAL TIMER Counter	Angle	+/-2°	6 Months

Table 10-2. Preset Pressure Control Settings

ITEM NO.	DESCRIPTION	PRESSURE SETTING
510	Relief Valve	85 PSIG
519	Back Pressure Regulator	3-9.5 PSI
520	Back Pressure Regulator	9-15 PSI
521	Filter/Regulator	90 PSIG
529	Pressure Regulator	10 PSIG
543	Pressure Regulator	20 PSIG
553	Purge Relief Valve	0.15 inch of water
E503	Purge Pressure Differential Switch	0.15 inch of water
E512	Facility Water Pressure Switch	40 PSIG
E513	Facility Air Pressure Switch	70 PSIG

10-3 CALIBRATION OF AUTOMATIC FLOW CONTROL AND BACK PRESSURE

This should be performed when master flow values do not match the operator requested values. This procedure performs an automatic recalibration of the Flow and Back pressure. This procedure requires the use of a calibration jig and takes approximately 30 minutes. Refer to Figures 4-1, 4-2, 5-1 and 5-2.

CAUTION

To prevent damage to equipment, this procedure should only be performed when approved by a qualified Test Engineer.

1. Ensure power is ON at electrical control cabinet of test stand.

2. Using clamping cylinder 541, install 3 inch straight pipe calibration jig in place of Unit Under Test (UUT).

3. Select UUT CODE 24.

4. Run system at maximum flow for 20 minutes to stabilize temperature.

5. Press PUMP/UUT POWER OFF pushbutton E16. POWER ON light E18 should extinguish.

6. Turn Test Mode selector switch E35 to STANDBY position.

7. Turn Keyswitch E41 to PROGRAM position.

8. On Terminal #2, E42 select CALIBRATION by pressing "MESSAGE/GROUP" key.

9. Enter "1" when requested for Yes/No response.

10. When requested, press PUMP/UUT POWER ON pushbutton E17. An automatic calibration will begin and last approximately 30 minutes. DO NOT INTERRUPT this process or results will be invalid.

11. When requested, press PUMP/UUT POWER OFF pushbutton E16. POWER ON light E18 should extinguish.

NOTE

You must SAVE DATA to permanently STORE your programming changes.

12. If results are valid, advance the GROUP to SAVE DATA and press "1" when requested to permanently save the changes.

13. If calibration is invalid, ignore all changes and return to normal test operation without saving the data.

14. Return to normal testing operation by turning Keyswitch E41 to RUN position.

QUANTUM DYNAMICS INC.

TURBINE FLOWMETER CALIBRATION

)EL NUMBER: QAF-12-WRL-1SC(TC)
₹IAL NUMBER: 1207

ST FIT STRAIGHT LINE Y=A+BX

X [HZ]	Y [GPM]	K [PULS/GAL]	Y(FIT) [GPM]	Y-Y(FIT) [GPM]
2.55000	0.10000	1530.00000	0.08403	0.01597
5.48000	0.21400	1536.44861	0.19535	0.01865
8.08000	0.31700	1529.33752	0.29413	0.02287
13.31000	0.51500	1550.67969	0.49284	0.02216
19.92000	0.76400	1564.39795	0.74397	0.02003
30.69000	1.15300	1597.05115	1.15315	-0.00015
40.54000	1.51100	1609.79480	1.52738	-0.01638
60.28000	2.25900	1601.06238	2.27736	-0.01836
80.59000	3.02800	1596.89551	3.04900	-0.02100
101.65000	3.82800	1593.26013	3.84913	-0.02113
149.74001	5.65900	1587.63037	5.67620	-0.01720
198.89999	7.52800	1585.28162	7.54393	-0.01593
301.09000	11.41600	1582.46313	11.42642	-0.01042
401.82001	15.24700	1581.24231	15.25345	-0.00645
500.91000	19.01800	1580.32397	19.01816	-0.00016
604.09998	22.94100	1579.96594	22.93864	0.00236
705.34998	26.79500	1579.43640	26.78542	0.00958
803.79999	30.53600	1579.38171	30.52582	0.01018
909.65997	34.55500	1579.49927	34.54774	0.00726
1011.72998	38.43000	1579.59399	38.42568	0.00432
1199.18005	45.54900	1579.63525	45.54745	0.00155
1424.72998	54.11600	1579.64001	54.11673	-0.00073
1711.37000	65.00000	1579.72620	65.00701	-0.00701

-0.01284930 +0.03799287X

'ANDARD ERROR OF ESTIMATE S= 0.01295
)RRELATION COEFFICIENT R= 0.999999762

APS
REC. INSP.
DATE APR 23 1990
JOB
AC
QTY 6
INSP
REJ
ITEM

· NOTE: THIS IS PRE-SUPERLINEARIZED DATA **

ITEM #523 (002)

Dick Munns Company

LIQUID AND GAS - FLOWMETER CALIBRATION SERVICE

Phone (213) 596-1559 10571 Calle Lee —133 Los Alamitos, CA 90720

Certificate

DATE 5-22-89

SUBMITTED FOR CALIBRATION BY Quantum Dynamics Co., Inc.

TYPE OF INSTRUMENT Turbine Meter

SERIAL NO. Q1519/1207

MODEL NO. QAF-12-WRL-1SC(TC)

MFGR. Q.D.

RATED ACCURACY Mnfrs. Spec.

CALIBRATION INTERVAL 12 months

INDICATED		ACTUAL		REMARKS	
C.P.S.	G.P.M.	C.P.S.	G.P.M.	C.P.S.	G.P.M.
2.55	.100	80.59	3.028	705.35	26.795
5.48	.214	101.65	3.828	803.80	30.536
8.08	.317	149.74	5.659	909.66	34.555
13.31	.515	198.90	7.528	1011.73	38.430
19.92	.764	301.09	11.416	1199.18	45.549
30.69	1.153	401.82	15.247	1424.73	54.116
40.54	1.511	500.91	19.018	1711.37	65.000
60.28	2.259	604.10	22.941		

CALIBRATED BY COMPARISON WITH THE FOLLOWING STANDARDS

Test units A-6, A-14 & A-33. N.B.S. Traceability #M-2055 & M-2293.

As per MIL-STD-45662. Fluid = MIL-C-7024 Type II @ 80°F.

ALL INSTRUMENTS USED TO ACCOMPLISH ABOVE CALIBRATION HAVE DIRECT TRACEABILITY TO THE NATIONAL BUREAU OF STANDARDS, WASHINGTON, D.C.

CALIBRATION BY RA/DM APPROVED BY S Munns

FINAL TEST CERTIFICATION

SERIAL NO: 1207

DATE 04-19-1990 10:09:37

S.G. 0.7670
WATER DENSITY BASED ON TEMPERATURE OF 60 deg. F

INP. FREQ. (HZ)	CORR. OUTPUT (GPM) (PPH)	IND. OUTPUT (GPM) (PPH)	ABS ERROR (GPM) (PPH)	% ERROR
2.55000	0.10000	0.10000	0.00000	0.0000%
	38.32545	38.32546	0.00000	
5.48000	0.21400	0.21400	0.00000	0.0000%
	82.01647	82.01648	0.00001	
8.08000	0.31700	0.31700	-0.00000	-0.0000%
	121.49169	121.49168	-0.00001	
13.31000	0.51500	0.51500	-0.00000	-0.0000%
	197.37608	197.37607	-0.00002	
19.92000	0.76400	0.76400	0.00000	0.0000%
	292.80649	292.80649	0.00000	
30.69000	1.15300	1.15300	0.00000	0.0000%
	441.89249	441.89249	0.00000	
40.54000	1.51100	1.51100	-0.00000	-0.0000%
	579.09766	579.09760	-0.00006	
60.28000	2.25900	2.25900	0.00000	0.0000%
	865.77203	865.77216	0.00012	
80.59000	3.02800	3.02740	-0.00060	-0.0198%
	1160.49487	1160.26489	-0.22998	
101.65000	3.82800	3.82800	-0.00000	-0.0000%
	1467.09839	1467.09827	-0.00012	
149.74001	5.65900	5.65870	-0.00030	-0.0053%
	2168.83740	2168.72241	-0.11499	
198.89999	7.52800	7.52810	0.00010	0.0013%
	2885.14014	2885.17847	0.03833	
301.09000	11.41600	11.41510	-0.00090	-0.0079%
	4375.23389	4374.88916	-0.34473	
401.82001	15.24700	15.24710	0.00010	0.0007%
	5843.48193	5843.52002	0.03809	

ITEM #523 (002)

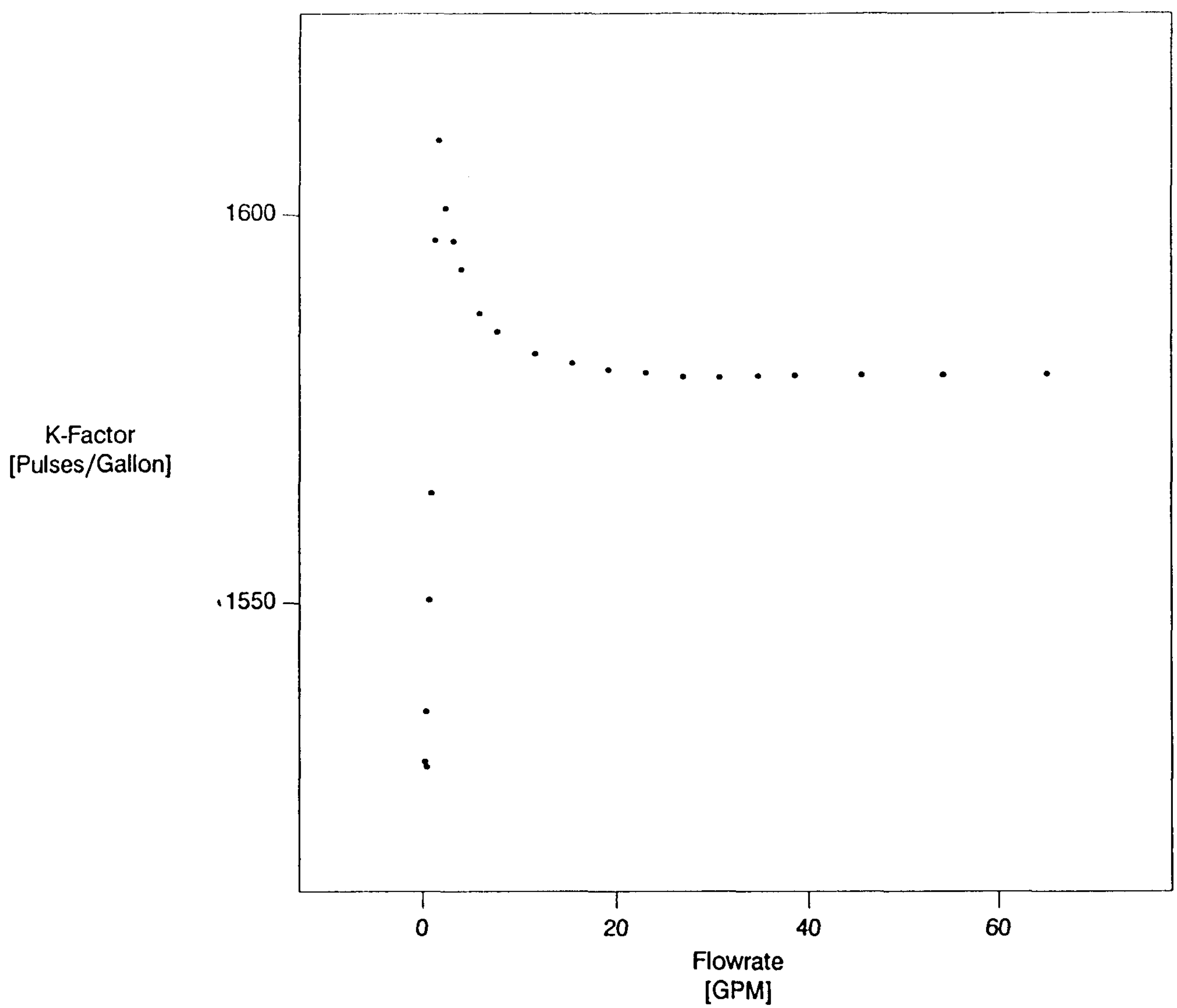
1207

INP.FREQ. (HZ)	CORR. OUTPUT (GPM) (PPH)	IND. OUTPUT (GPM) (PPH)	ABS. ERROR (GPM) (PPH)	% ERROR
500.91000	19.01800	19.01920	0.00120	0.0063%
	7288.73486	7289.19482	0.45996	
604.09998	22.94100	22.93900	-0.00200	-0.0087%
	8792.24219	8791.47461	-0.76758	
705.34998	26.79500	26.79810	0.00310	0.0116%
	10269.30566	10270.49414	1.18848	
803.79999	30.53600	30.53150	-0.00450	-0.0147%
	11703.06055	11701.33789	-1.72266	
909.65997	34.55500	34.55600	0.00100	0.0029%
	13243.36133	13243.74414	0.38281	
1011.72998	38.43000	38.43100	0.00100	0.0026%
	14728.47266	14728.85547	0.38281	
1199.18005	45.54900	45.55100	0.00200	0.0044%
	17456.86133	17457.62695	0.76563	
1424.72998	54.11600	54.12000	0.00399	0.0074%
	20740.20313	20741.73438	1.53125	
1711.37000	65.00000	65.00260	0.00260	0.0040%
	24911.54492	24912.54297	0.99805	

CERTIFIED BY : ______________________

APS
REC. INSP.
DATE APR 23 1990
JOB P
AC REL
QTY 8 ITEM
INSP

K-FACTOR PLOT: D1207.k



Q1519

ITEM #523 (002)

QUANTUM DYNAMICS INC.

RTD TEMPERATURE TRANAMITTER CALIBRATION

MODEL NO. 2B59A-0-1-02

SERIAL NO. 8905

TEMPERATURE RANGE: +20 F -- +120 F

OUTPUT RANGE: 4 -- 20 MA

TEMPERATURE(F)	RESISTANCE(OHM)	OUTPUT(MA)
20	97.26	4.00
32	100.00	5.97
68	107.79	11.58
104	115.54	17.20
120	119.39	20.00

ACCEPTED AND CERTIFIED BY: [signature]

DATE: May 2. 89

QUANTUM DYNAMICS INC.

TURBINE FLOWMETER CALIBRATION

IDEL NUMBER: QAF-32-WRL-1SC(TC)
:RIAL NUMBER: 1209

:ST FIT STRAIGHT LINE Y=A+BX

I	X [HZ]	Y [GPM]	K [PULS/GAL]	Y(FIT) [GPM]	Y-Y(FIT) [GPM]
.	1.00000	0.95900	62.56517	0.90457	0.05443
;	1.27000	1.20000	63.50000	1.14334	0.05666
3	1.76000	1.60000	66.00000	1.57666	0.02334
1	2.20000	2.00000	66.00000	1.96576	0.03424
5	2.69000	2.41800	66.74938	2.39909	0.01891
5	3.93000	3.50100	67.35219	3.49565	0.00535
'	5.17000	4.58600	67.64065	4.59222	-0.00622
3	8.17000	7.24200	67.68848	7.24520	-0.00320
)	11.35000	10.04100	67.82193	10.05737	-0.01637
)	15.12000	13.36300	67.88895	13.39128	-0.02828
L	20.30000	17.92800	67.93842	17.97210	-0.04410
2	24.59000	21.71500	67.94382	21.76587	-0.05087
3	30.21000	26.68400	67.92834	26.73579	-0.05179
4	40.07000	35.42700	67.86349	35.45526	-0.02826
5	60.15000	53.18500	67.85748	53.21257	-0.02757
6	80.57000	71.23900	67.85890	71.27054	-0.03154
7	105.58000	93.34400	67.86510	93.38758	-0.04358
8	134.89999	119.36700	67.80769	119.31607	0.05093
9	150.70000	133.34000	67.81161	133.28845	0.05154
0	203.69000	180.20000	67.82131	180.14899	0.05101
1	265.72000	235.05000	67.82897	235.00385	0.04616
2	340.50000	301.15100	67.83972	301.13388	0.01712
3	373.23001	330.00000	67.86000	330.07794	-0.07794

= 0.02024313 +0.88432789X

TANDARD ERROR OF ESTIMATE S= 0.04689
ORRELATION COEFFICIENT R= 0.999999881

* NOTE: THIS IS PRE-SUPERLINEARIZED DATA **

ITEM #E524 (002)

Dick Munns Company

LIQUID AND GAS - FLOWMETER CALIBRATION SERVICE

Phone (213) 596-1559   10571 Calle Lee —133   Los Alamitos, CA 90720

Certificate

DATE 5-22-89

SUBMITTED FOR CALIBRATION BY Quantum Dynamics Co., Inc.

TYPE OF INSTRUMENT Turbine Meter

SERIAL NO. Q1519/1209

MODEL NO. QAF-32-WRL-1SC(TC)

MFGR. Q.D.

RATED ACCURACY Mnfrs. Spec.

CALIBRATION INTERVAL 12 months

INDICATED		ACTUAL		REMARKS	
C.P.S.	G.P.M.	C.P.S.	G.P.M.	C.P.S.	G.P.M.
1.00	.959	11.35	10.041	105.58	93.344
1.27	1.200	15.12	13.363	134.90	119.367
1.76	1.600	20.30	17.928	150.70	133.340
2.20	2.000	24.59	21.715	203.69	180.200
2.69	2.418	30.21	26.684	265.72	235.050
3.93	3.501	40.07	35.427	340.50	301.151
5.17	4.586	60.15	53.185	373.23	330.000
8.17	7.242	80.57	71.239		

CALIBRATED BY COMPARISON WITH THE FOLLOWING STANDARDS.

Test units A-6, A-14 & A-33. N.B.S. Traceability#M-2055 & M-2293.

As per MIL-STD-45662. Fluid = MIL-C-7024 Type II @ 80°F.

ALL INSTRUMENTS USED TO ACCOMPLISH ABOVE CALIBRATION HAVE DIRECT TRACEABILITY TO THE NATIONAL BUREAU OF STANDARDS, WASHINGTON, D. C.

CALIBRATION BY RA/DM   APPROVED BY S/ Munns

FINAL TEST CERTIFICATION

SERIAL NO: 1209

DATE 04-19-1990 10:01:18

S.G. 0.7670
WATER DENSITY BASED ON TEMPERATURE OF 60 deg. F

APS
REC. INSP.
DATE APR 23 1990
JOB
AC
QTY
INSP
REL
ITEM
AP
6

INP.FREQ. (HZ)	CORR. OUTPUT (GPM) (PPH)	IND. OUTPUT (GPM) (PPH)	ABS. ERROR (GPM) (PPH)	% ERROR
1.00000	0.95900 367.54111	0.95900 367.54111	0.00000 0.00000	0.0000%
1.27000	1.20000 459.90549	1.20000 459.90543	-0.00000 -0.00006	-0.0000%
1.76000	1.60000 613.20728	1.60000 613.20734	0.00000 0.00006	0.0000%
2.20000	2.00000 766.50909	2.00000 766.50909	0.00000 0.00000	0.0000%
2.69000	2.41800 926.70947	2.41800 926.70941	-0.00000 -0.00006	-0.0000%
3.93000	3.50100 1341.77417	3.50100 1341.77417	0.00000 0.00000	0.0000%
5.17000	4.58600 1757.60535	4.58600 1757.60510	-0.00000 -0.00024	-0.0000%
8.17000	7.24200 2775.52954	7.24200 2775.52930	-0.00000 -0.00024	-0.0000%
11.35000	10.04100 3848.25903	10.04100 3848.25903	0.00000 0.00000	0.0000%
15.12000	13.36300 5121.43066	13.36300 5121.43066	0.00000 0.00000	0.0000%
20.30000	17.92800 6870.98730	17.92880 6871.29443	0.00080 0.30713	0.0045%
24.59000	21.71500 8322.37207	21.71760 8323.37012	0.00260 0.99805	0.0120%
30.21000	26.68400 10226.76465	26.68150 10225.80566	-0.00250 -0.95898	-0.0094%
40.07000	35.42700 13577.55859	35.42811 13577.98340	0.00111 0.42480	0.0031%

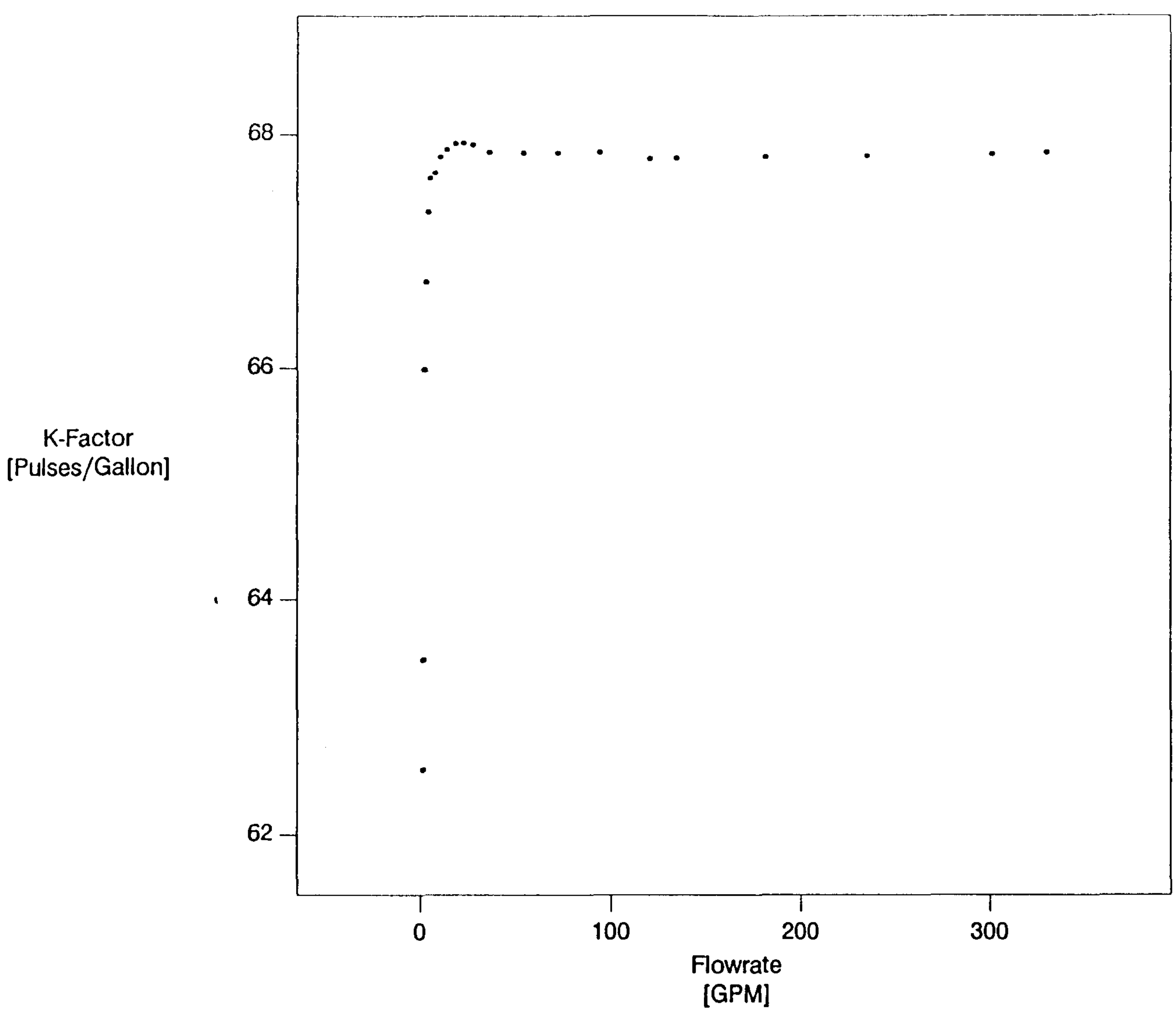
INP.FREQ. (HZ)	CORR. OUTPUT (GPM) (PPH)	IND. OUTPUT (GPM) (PPH)	ABS. ERROR (GPM) (PPH)	% ERROR
60.15000	53.18500	53.18010	-0.00491	-0.0092%
	20383.39453	20381.51367	-1.88086	
80.57000	71.23900	71.22640	-0.01260	-0.0177%
	27302.66992	27297.84180	-4.82813	
105.58000	93.34400	93.34850	0.00449	0.0048%
	35774.51172	35776.23438	1.72266	
134.89999	119.36700	119.39199	0.02499	0.0209%
	45747.94531	45757.52344	9.57813	
150.70000	133.34000	133.32831	-0.01169	-0.0088%
	51103.16016	51098.67969	-4.48047	
203.69000	180.20000	180.20569	0.00569	0.0032%
	69062.46875	69064.64844	2.17969	
265.72000	235.05000	235.10310	0.05310	0.0226%
	90083.98438	90104.33594	20.35156	
340.50000	301.15100	301.16379	0.01279	0.0042%
	115417.49219	115422.39063	4.89844	
373.23001	330.00000	330.01007	0.01007	0.0031%
	126474.00000	126477.85938	3.85938	

CERTIFIED BY : ____________________

APS
REC. INSP.
DATE APR 23 1990
JOB
REJ
AC
ITEM
QTY 8
INSP

ITEM #E524 (002)

K-FACTOR PLOT: D1209.k



ITEM #E524 (002) Q1519

QUANTUM DYNAMICS INC.

RTD TEMPERATURE TRANAMITTER CALIBRATION

MODEL NO. 2B59A-0-1-02

SERIAL NO. 8907

TEMPERATURE RANGE: +20 F -- +120 F

OUTPUT RANGE: 4 -- 20 MA

TEMPERATURE(F)	RESISTANCE(OHM)	OUTPUT(MA)
20	97.26	4.00
32	100.00	5.97
68	107.79	11.58
104	115.54	17.19
120	119.39	20.00

ACCEPTED AND CERTIFIED BY: [signature]

DATE: May 2. 89

ITEM #E529. (002)

Viatran Performance Certificate

	04-18-1991
L	108S70A
.NO.	14435141
ENCE	1082AK2S70A0
	10 VDC
IT	2.610 MV/V
	0-75 PSIG

CALE PRESSURE RANGE

WIRING CONNECTIONS

RED	+ INPUT
GREEN	- INPUT
BLACK	+ OUTPUT
WHITE	- OUTPUT

	STANDARD CALIBRATION		SPECIAL CALIBRATION		
% FSPR	RUN 1	RUN 2		RUN 1	RUN 2
INCREASING 0	+0.12				
20					
40					
60					
80					
100	26.22				
DECREASING 80					
60					
40					
20					
0	+0.12				

:S:

ATION CIRCUIT

ntage of Viatran strain gage pressure sensors is that the gain or span adout system can be set without applying a known pressure input. The setup can be accomplished by using a calibrate circuit as explained

AL CIRCUIT

nsor models are supplied with an internal shunt calibrate circuit either ndard or by special order. If this sensor has the circuit included, it will l below. Simply shorting the proper pins will produce the calibration ignal listed.

IAL CIRCUIT

nsor does not include the calibration circuit, the same type calibration accomplished by connecting a resistor of the value indicated across er pins. This technique is accurate for cable lengths of up to 200 ft.

ansducer - Internal Calibration

norting pins & will produce calibration output signal.

ansducer - External Calibration

onnecting ohms across ns & will produce calibration output signal.

ansmitter* - Internal Calibration Circuit and Switch

il 1 represents with a reading

il 2 represents with a reading

OTE: All calibrations should be performed only after the transducer has been adjusted for zero at zero pressure.

WARRANTY

Viatran Corporation warrants that its products shall be free from defective parts and workmanship for a period of twelve (12) months from date of original shipment, provided that Viatran's obligation hereunder shall be limited to correcting any defective workmanship and/or replacing any defective material F.O.B. destination.

A repair is warranted ninety (90) days from repair date under conditions of original warranty period unless superseded by original warranty period. If inspection by the Company of such product does not disclose any defect of workmanship or material, the Company's regular charges will apply. This warranty carries no liability, either expressed or implied, beyond our obligation to replace the unit which carries the warranty. This warranty is in lieu of all other warranties of merchantability or fitness. No allowance will be made for any expense incurred for correcting any defective workmanship and/or material without written consent by Viatran. Unit must be shipped to the Company, transportation prepaid, and return authorization number must be referenced on the package to assure acceptance at our shipping dock. Prices, specifications and decisions subject to change without notice.

The Company shall not be liable for and the Purchaser assumes and agrees to indemnify and save harmless the Company in respect to any loss or damage that may arise through the use by the Purchaser, or others, of any of the Company's products.

This warranty is void if the product is subjected to misuse, accident, neglect or improper application, installation or operation. This warranty is void if prior defects in materials or workmanship repairs are made by anyone except Viatran or its authorized service agency.

REPAIR

Most Viatran sensors have been designed to be easily repaired and recalibrated if necessary. If a failure occurs, the sensor should be returned to the factory for inspection and testing. If the sensor failure is covered by our one year warranty policy, the unit will be repaired as necessary and reshipped without delay. Sensors usually not covered by warranty can be repaired within four to six weeks for approximately thirty to sixty per cent of the purchase price. Simple repairs can often be made for minimum charge. Units should be returned to the attention of the Repair Department, after obtaining a return authorization number from the customer service desk at (716) 773-1700.

QUALITY ASSURANCE

V 4

Viatran CORPORATION
300 Industrial Drive Grand Island, New York 14072
Telephone (716) 773-1700 TWX # 710-260-1353
FAX (716) 773-2488

ITEM #E530 (002)

Viatran Performance Certificate

	04-09-1991
L	120S70A
.NO.	12817059
ENCE	1202AF5S70A0
	10 VDC
IT	MVDC
	+/-30 PSID
CALE PRESSURE RANGE	

WIRING CONNECTIONS

PIN 1	+ INPUT
PIN 2	- INPUT
PIN 3	+ OUTPUT
PIN 4	- OUTPUT

	STANDARD CALIBRATION			SPECIAL CALIBRATION	
% FSPR	RUN 1	RUN 2		RUN 1	RUN 2
INCREASING 0	HI -0.22	LO -0.22			
20	2.67	-3.12			
40	5.56	-6.01			
60	8.45	-8.90			
80	11.33	-11.79			
100	14.19	-14.65			
DECREASING 80					
60	8.45	-8.90			
40					
20					
0	-0.22	-0.22			

:S:

ALIBRATION PRESSURE= 24.08 PSID

ATION CIRCUIT

ntage of Viatran strain gage pressure sensors is that the gain or span adout system can be set without applying a known pressure input. The setup can be accomplished by using a calibrate circuit as explained

AL CIRCUIT

isor models are supplied with an internal shunt calibrate circuit either ıdard or by special order. If this sensor has the circuit included, it will l below. Simply shorting the proper pins will produce the calibration gnal listed.

AL CIRCUIT

isor does not include the calibration circuit, the same type calibration accomplished by connecting a resistor of the value indicated across er pins. This technique is accurate for cable lengths of up to 200 ft.

ansducer - Internal Calibration

ıorting pins & will produce

calibration output signal.

ansducer - External Calibration

onnecting 75500 ohms across

ıs 2 & 4 will produce

11.37 MV calibration output signal.

ansmitter* - Internal Calibration Circuit and Switch

il 1 represents with a reading

il 2 represents with a reading

)TE: All calibrations should be performed only after the transducer

WARRANTY

Viatran Corporation warrants that its products shall be free from defective parts and workmanship for a period of twelve (12) months from date of original shipment, provided that Viatran's obligation hereunder shall be limited to correcting any defective workmanship and/or replacing any defective material F.O.B. destination.

A repair is warranted ninety (90) days from repair date under conditions of original warranty period unless superseded by original warranty period. If inspection by the Company of such product does not disclose any defect of workmanship or material, the Company's regular charges will apply. This warranty carries no liability, either expressed or implied, beyond our obligation to replace the unit which carries the warranty. This warranty is in lieu of all other warranties of merchantability or fitness. No allowance will be made for any expense incurred for correcting any defective workmanship and/or material without written consent by Viatran. Unit must be shipped to the Company, transportation prepaid, and return authorization number must be referenced on the package to assure acceptance at our shipping dock. Prices, specifications and decisions subject to change without notice.

The Company shall not be liable for and the Purchaser assumes and agrees to indemnify and save harmless the Company in respect to any loss or damage that may arise through the use by the Purchaser, or others, of any of the Company's products.

This warranty is void if the product is subjected to misuse, accident, neglect or improper application, installation or operation. This warranty is void if prior defects in materials or workmanship repairs are made by anyone except Viatran or its authorized service agency.

REPAIR

Most Viatran sensors have been designed to be easily repaired and recalibrated if necessary. If a failure occurs, the sensor should be returned to the factory for inspection and testing. If the sensor failure is covered by our one year warranty policy, the unit will be repaired as necessary and reshipped without delay. Sensors usually not covered by warranty can be repaired within four to six weeks for approximately thirty to sixty per cent of the purchase price. Simple repairs can often be made for minimum charge. Units should be returned to the attention of the Repair Department, after obtaining a return authorization number from the customer service desk at (716) 773-1700.

QUALITY ASSURANCE

Viatran CORPORATION

300 Industrial Drive Grand Island, New York 14072
Telephone (716) 773-1700 TWX # 710-260-1353
FAX (716) 773-2488

V4

ITEM # E 531 (002)

Viatran Performance Certificate

	05-04-1989
-	108S70/1109
. NO.	12807359
ENCE	1082AK2AS700/12131
	110 VAC 60 HZ
IT	DIGITAL READOUT
	0-75 PSIG
CALE PRESSURE RANGE	

WIRING CONNECTIONS
+ INPUT
- INPUT
+ OUTPUT
- OUTPUT

	% FSPR	STANDARD CALIBRATION RUN 1	STANDARD CALIBRATION RUN 2	SPECIAL CALIBRATION	RUN 1	RUN 2
INCREASING	0	0.00 CTS.				
	20	15.06 CTS.				
	40	30.10 CTS.				
	60	45.12 CTS.				
	80	60.10 CTS.				
	100	75.00 CTS.				
DECREASING	80					
	60	45.12 CTS.				
	40			* NOTE *		
	20			CTS.=COUNTS		
	0	0.00 CTS.				

:ALIBRATION READING OF 60.08 COUNTS @ 59.98 PSIG READ THRU INDICATOR.
IODEL 108S70 WITH INDICATING SYSTEM.

ATION CIRCUIT
ntage of Viatran strain gage pressure sensors is that the gain or span adout system can be set without applying a known pressure input. The setup can be accomplished by using a calibrate circuit as explained

AL CIRCUIT
isor models are supplied with an internal shunt calibrate circuit either idard or by special order. If this sensor has the circuit included, it will below. Simply shorting the proper pins will produce the calibration gnal listed.

AL CIRCUIT
isor does not include the calibration circuit, the same type calibration accomplished by connecting a resistor of the value indicated across er pins. This technique is accurate for cable lengths of up to 200 ft.

ansducer - Internal Calibration
iorting pins & will produce
calibration output signal.

ansducer - External Calibration
onnecting ohms across
is & will produce
calibration output signal.

ansmitter* - Internal Calibration Circuit and Switch
il 1 represents with a reading

.l 2 represents with a reading

)TE: All calibrations should be performed only after the transducer

WARRANTY

Viatran Corporation warrants that its products shall be free from defective parts and workmanship for a period of twelve (12) months from date of original shipment, provided that Viatran's obligation hereunder shall be limited to correcting any defective workmanship and/or replacing any defective material F.O.B. destination.

A repair is warranted ninety (90) days from repair date under conditions of original warranty period unless superseded by original warranty period. If inspection by the Company of such product does not disclose any defect of workmanship or material, the Company's regular charges will apply. This warranty carries no liability, either expressed or implied, beyond our obligation to replace the unit which carries the warranty. This warranty is in lieu of all other warranties of merchantability or fitness. No allowance will be made for any expense incurred for correcting any defective workmanship and/or material without written consent by Viatran. Unit must be shipped to the Company, transportation prepaid, and return authorization number must be referenced on the package to assure acceptance at our shipping dock. Prices, specifications and decisions subject to change without notice.

The Company shall not be liable for and the Purchaser assumes and agrees to indemnify and save harmless the Company in respect to any loss or damage that may arise through the use by the Purchaser, or others, of any of the Company's products.

This warranty is void if the product is subjected to misuse, accident, neglect or improper application, installation or operation. This warranty is void if prior defects in materials or workmanship repairs are made by anyone except Viatran or its authorized service agency.

REPAIR

Most Viatran sensors have been designed to be easily repaired and recalibrated if necessary. If a failure occurs, the sensor should be returned to the factory for inspection and testing. If the sensor failure is covered by our one year warranty policy, the unit will be repaired as necessary and reshipped without delay. Sensors usually not covered by warranty can be repaired within four to six weeks for approximately thirty to sixty per cent of the purchase price. Simple repairs can often be made for minimum charge. Units should be returned to the attention of the Repair Department, after obtaining a return authorization number from the customer service desk at (716) 773-1700.

QUALITY ASSURANCE

Viatran CORPORATION
300 Industrial Drive Grand Island New York 14072
Telephone (716) 773-1700 TWX # 710-260-1353

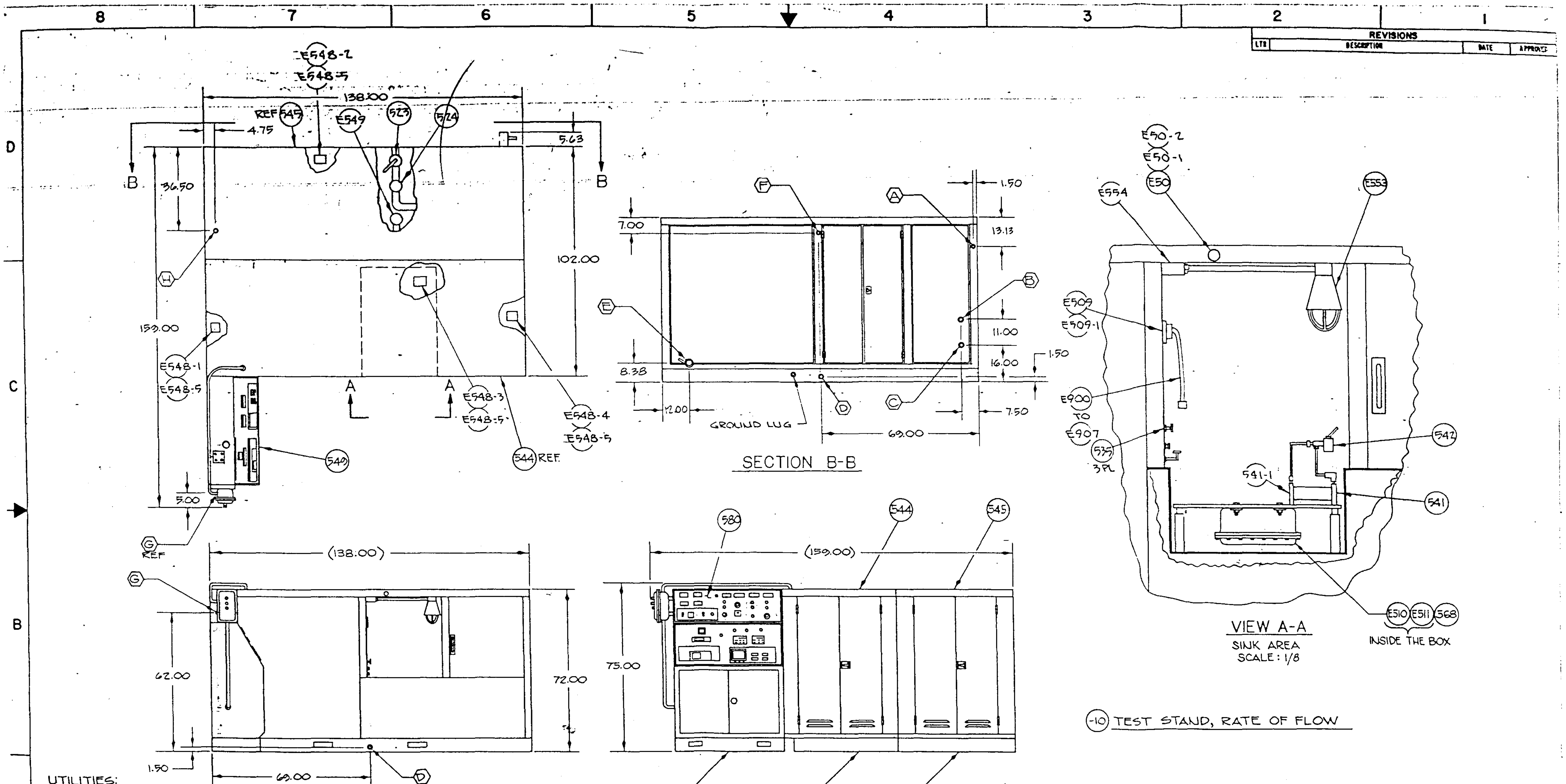
CHAPTER XI

SYSTEM DRAWINGS

11-1 GENERAL

The drawings in this chapter are provided to aid in the operation, troubleshooting, and repair of the Rate of Flow Test Stand. Hydraulic, pneumatic and electrical schematics are placed at the end of Chapter IV. Table 11-1 provides an index of drawings which follow.

Table 11-1. Drawings Index



UTILITIES:

QTY.	IDENT.	SIZE DESCRIPTION
1	A	.50NPT, FACILITY AIR 50SCFM AT 100 PSI
1	B	1.50NPT, COOLING WATER OUTLET
1	C	1.50NPT, COOLING WATER INLET
2	D	1.00NPT, DRIP PAN DRAIN
1	E	2.00NPT, EMERGENCY RSVR. DRAIN
1	F	1.00NPT, PRESSURE FILL.
1	G	.50NPT CONDUIT, 115VAC, 60HZ, 1Ø, 40AMPS, CONTROL POWER
1	H	1.50NPT CONDUIT, 480VAC, 60HZ., 3Ø, 50AMPS, FACILITY POWER

BASE:
SHOULD BE SMOOTH AND LEVEL TO THE FLOOR.

WEIGHT:
TOTAL WT. — 12,700LBS. (APPROX.)

SEE SEPARATE P/L 10047 & 20071

PARTS LIST

CONTRACT NO. F41608-88-C-2452

APS Systems 2535 W FIFTH STREET OXNARD, CA. 93030, USA

ASSY. AND INSTALATION TEST STAND, RATE OF FLOW

D 60984 10047

FINAL RATE OF FLOW

APPLICATION

Figure 11-1. Assembly and Installation (Sheet 1 of 2)

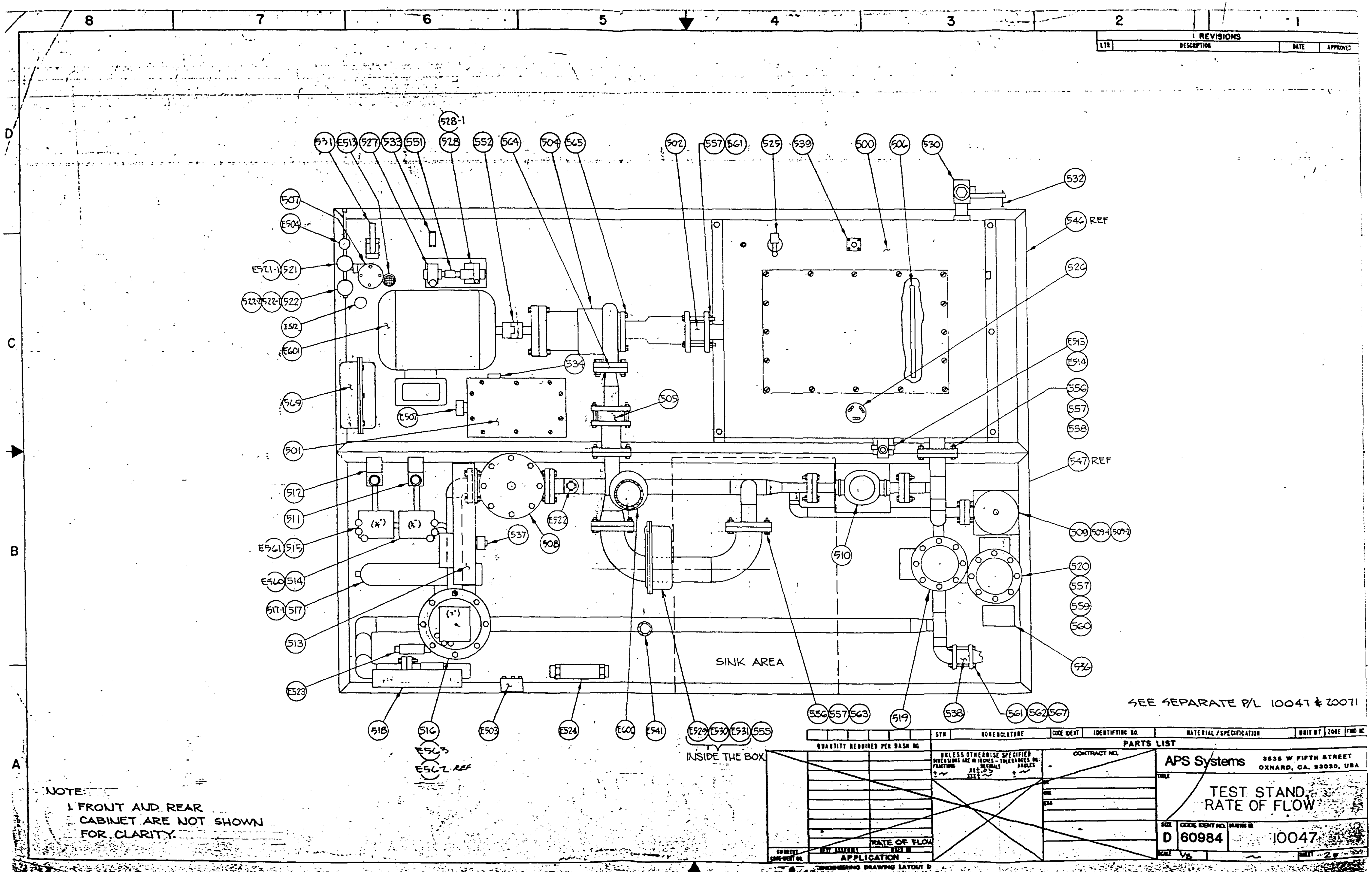


Figure 11-1. Assembly and Installation
(Sheet 2 of 2)

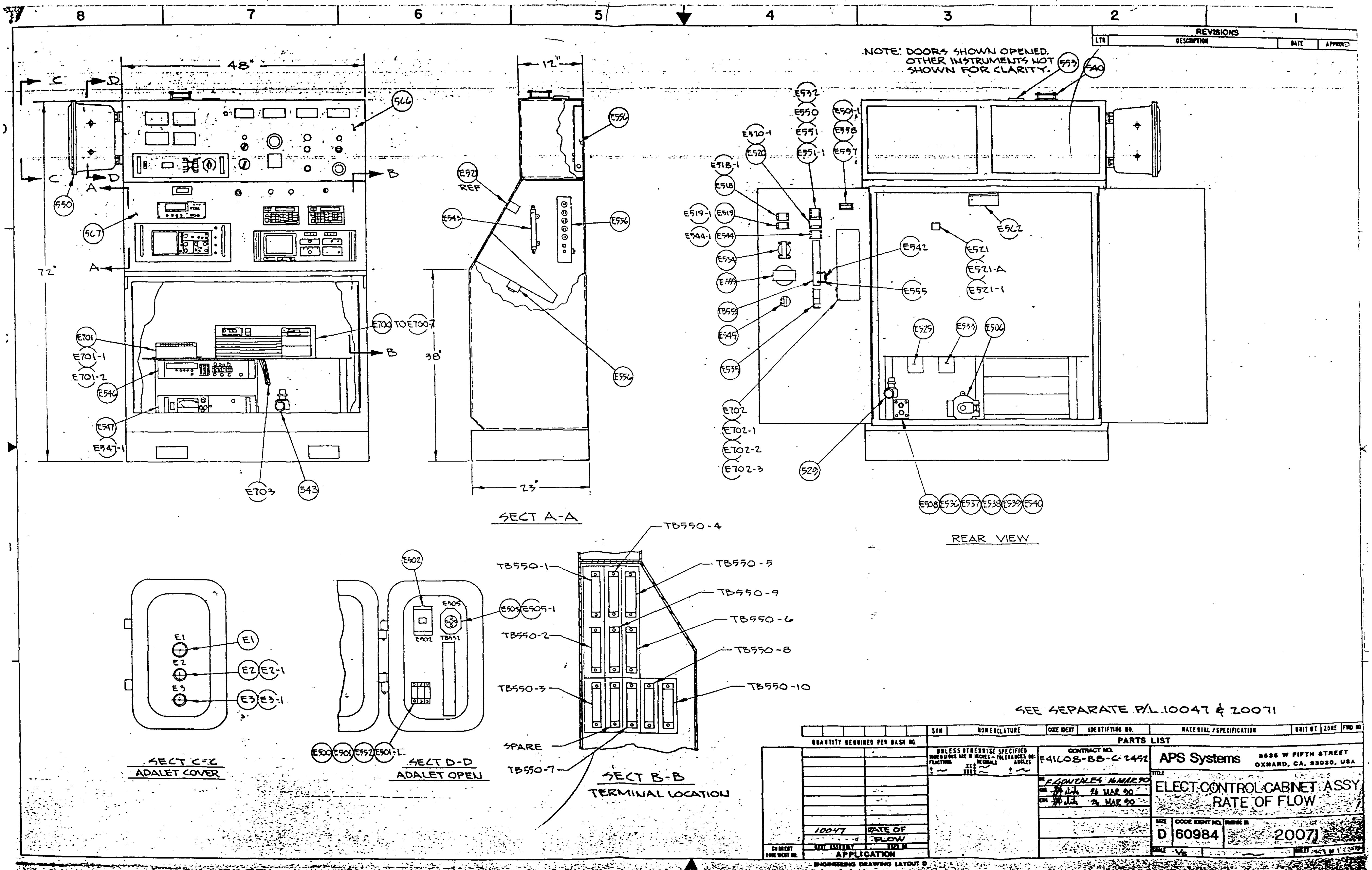


Figure 11-2. Electrical Control Cabinet Assembly

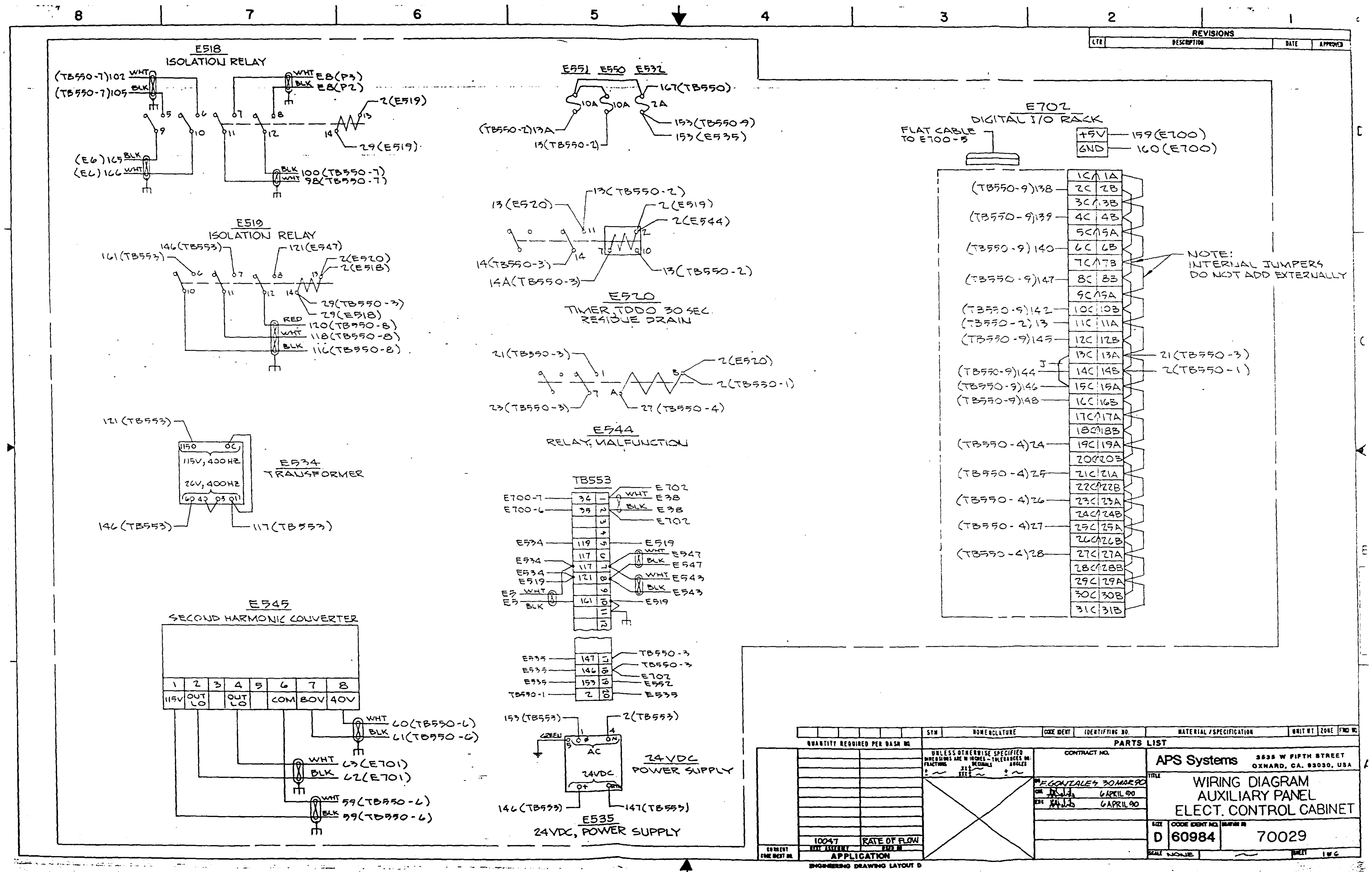


Figure 11-3. Wiring Diagram Electrical Control Cabinet (Sheet 1 of 6)

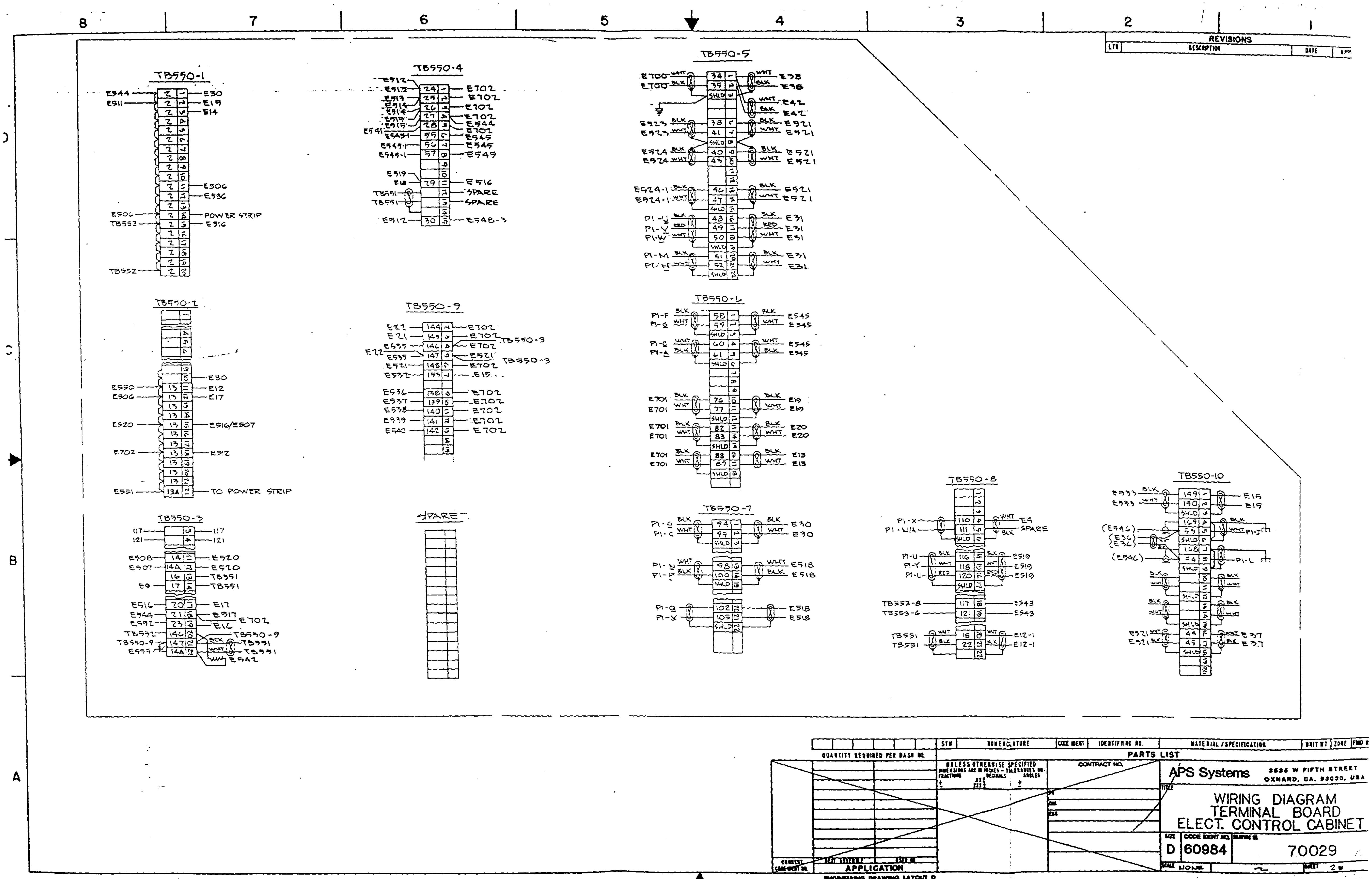
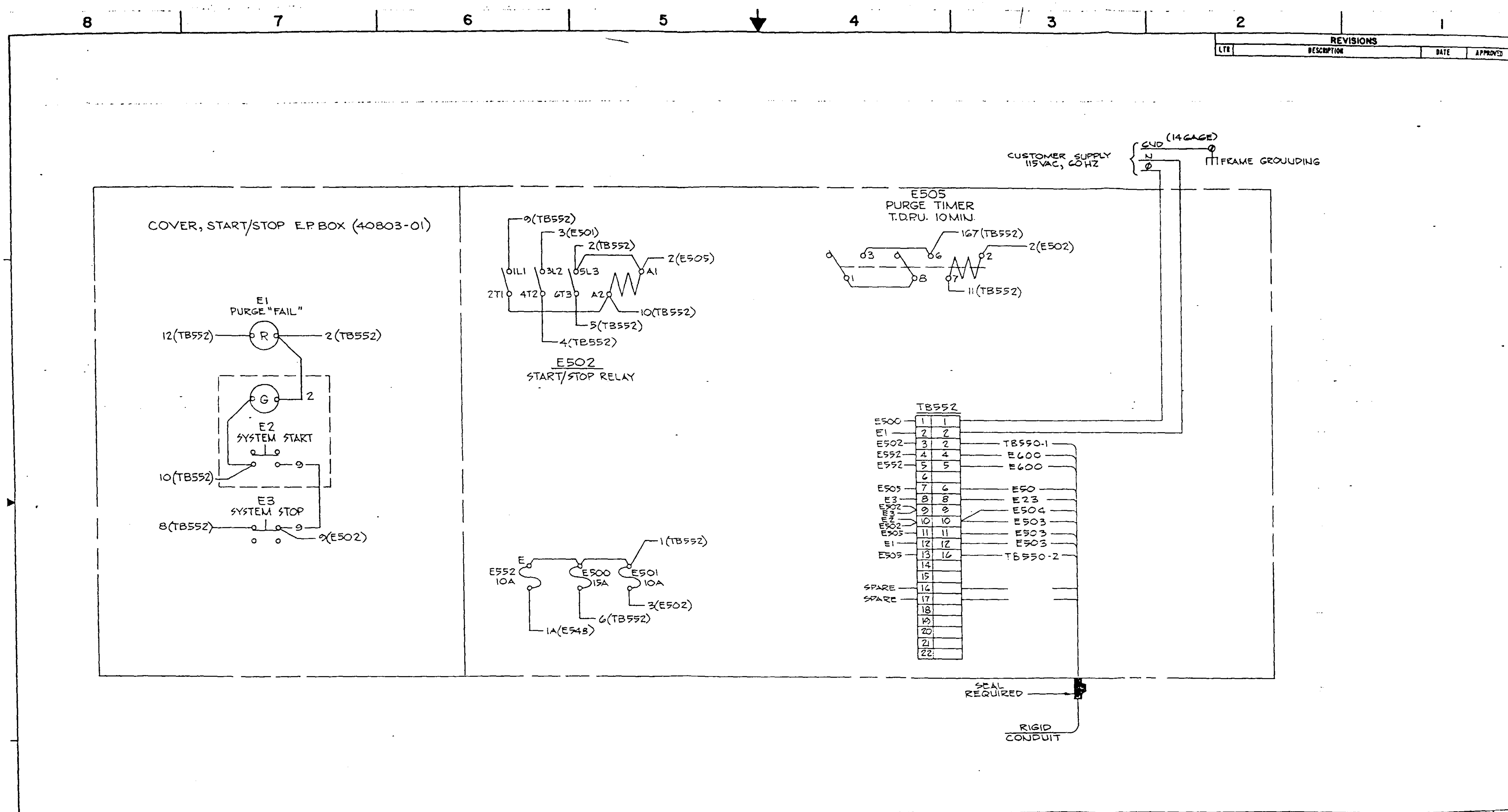


Figure 11-3. Wiring Diagram Electrical Control Cabinet (Sheet 2 of 6)



	PARTS LIST
APS Systems	3525 W FIFTH STREET OXNARD, CA. 93030, USA
TITLE	WIRING DIAGRAM START/STOP EP BOX ELECT. CONTROL CABINET
SIZE D	CODE IDENT NO. 60984 / DWG NO. 70029
SCALE NONE	SHEET 3

APPLICATION

ENGINEERING DRAWING LAYOUT D

Figure 11-3. Wiring Diagram Electrical Control Cabinet (Sheet 3 of 6)

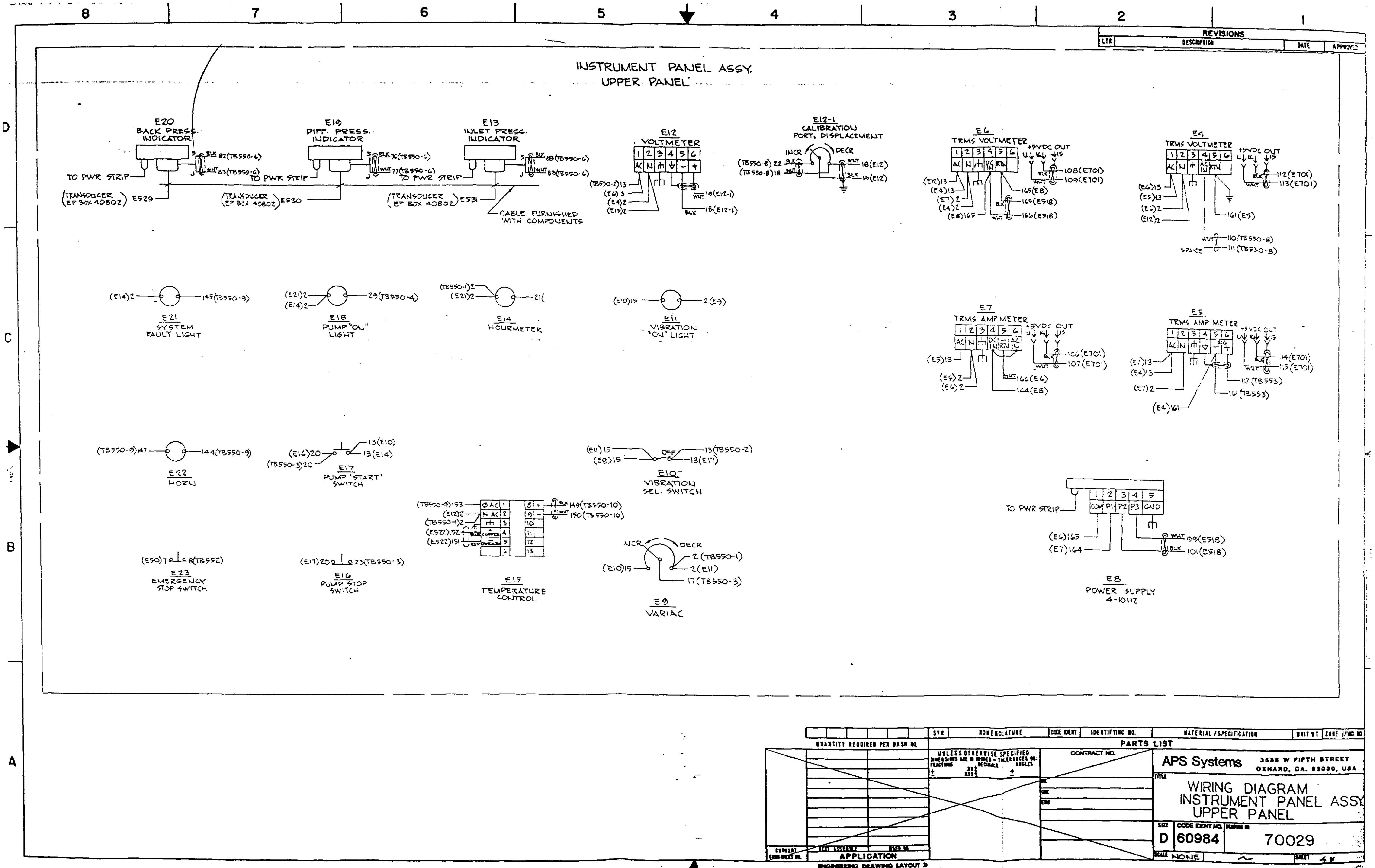


Figure 11-3. Wiring Diagram Electrical Control Cabinet (Sheet 4 of 6)

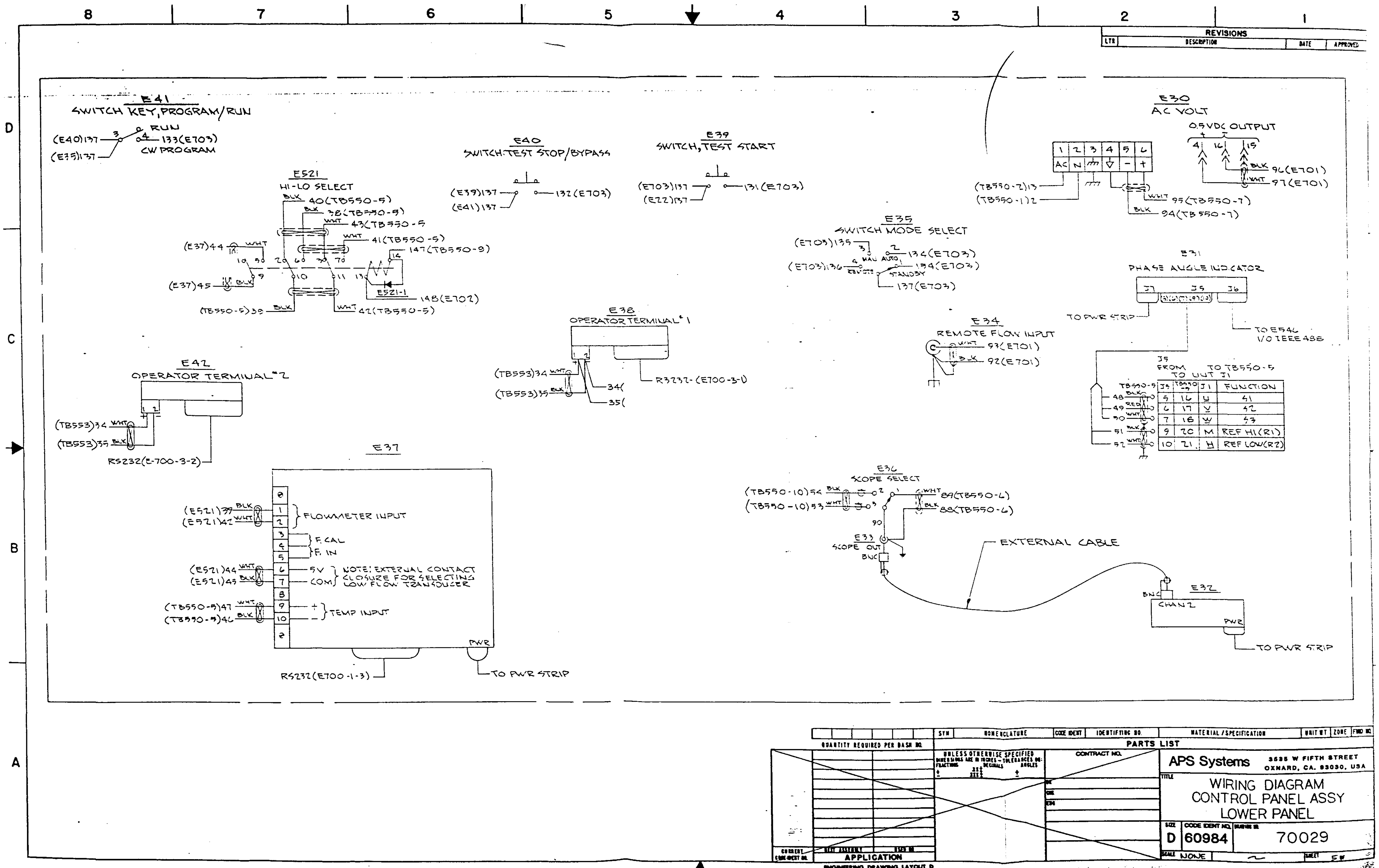


Figure 11-3. Wiring Diagram Electrical Control Cabinet (Sheet 5 of 6)

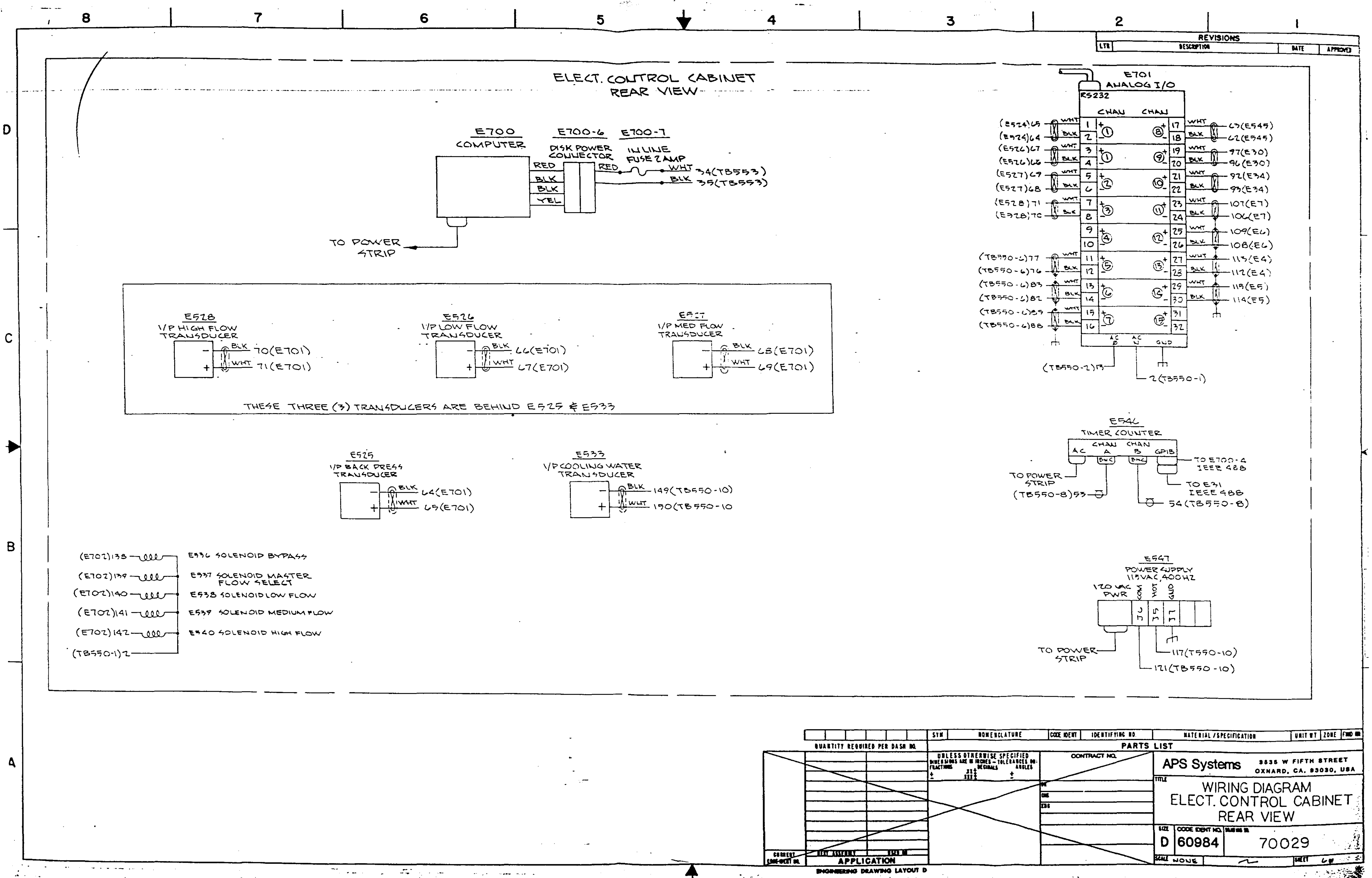


Figure 11-3. Wiring Diagram Electrical Control Cabinet (Sheet 6 of 6)

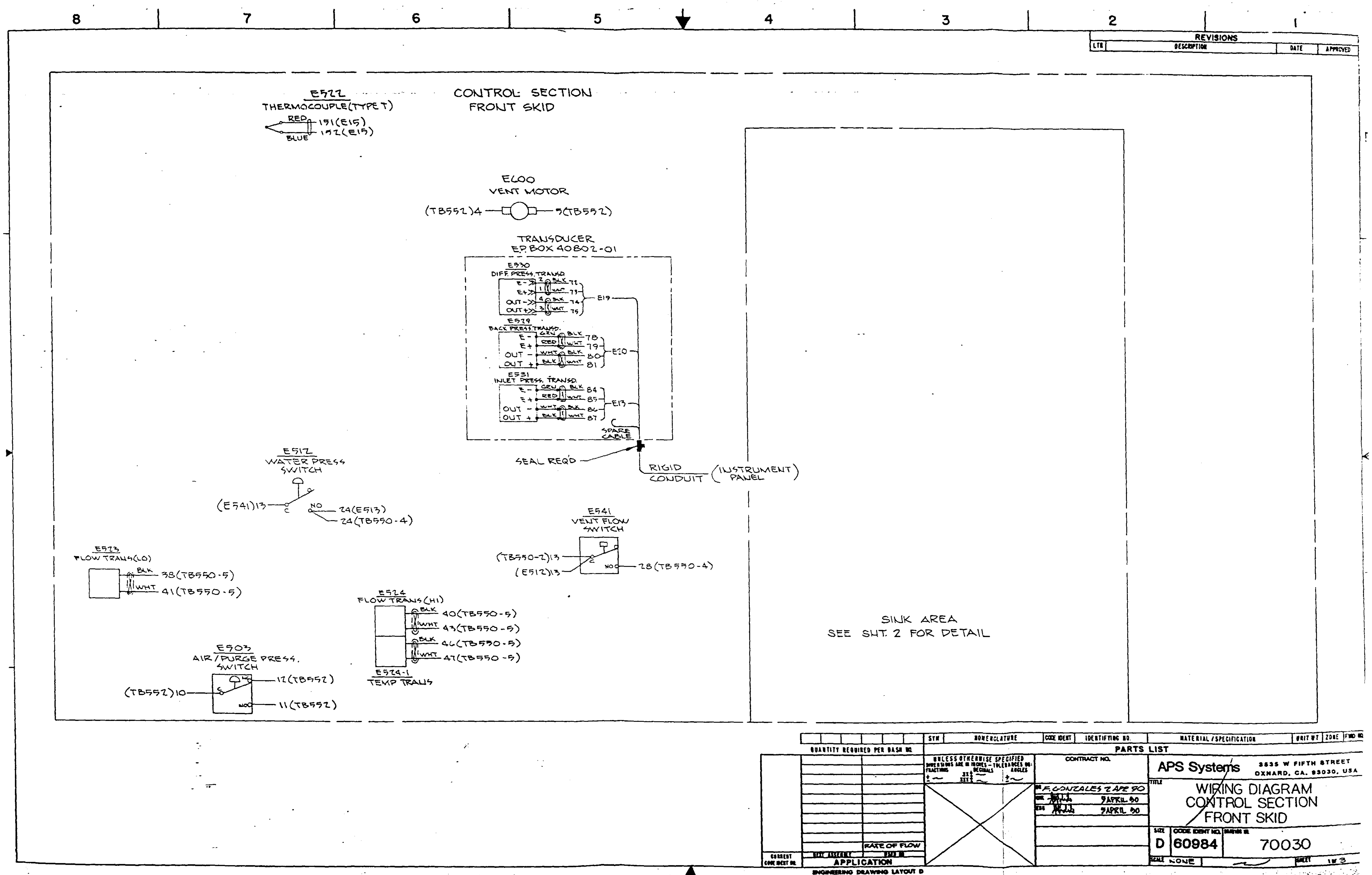


Figure 11-4. Wiring Diagram Test-Stand (Sheet 1 of 3)

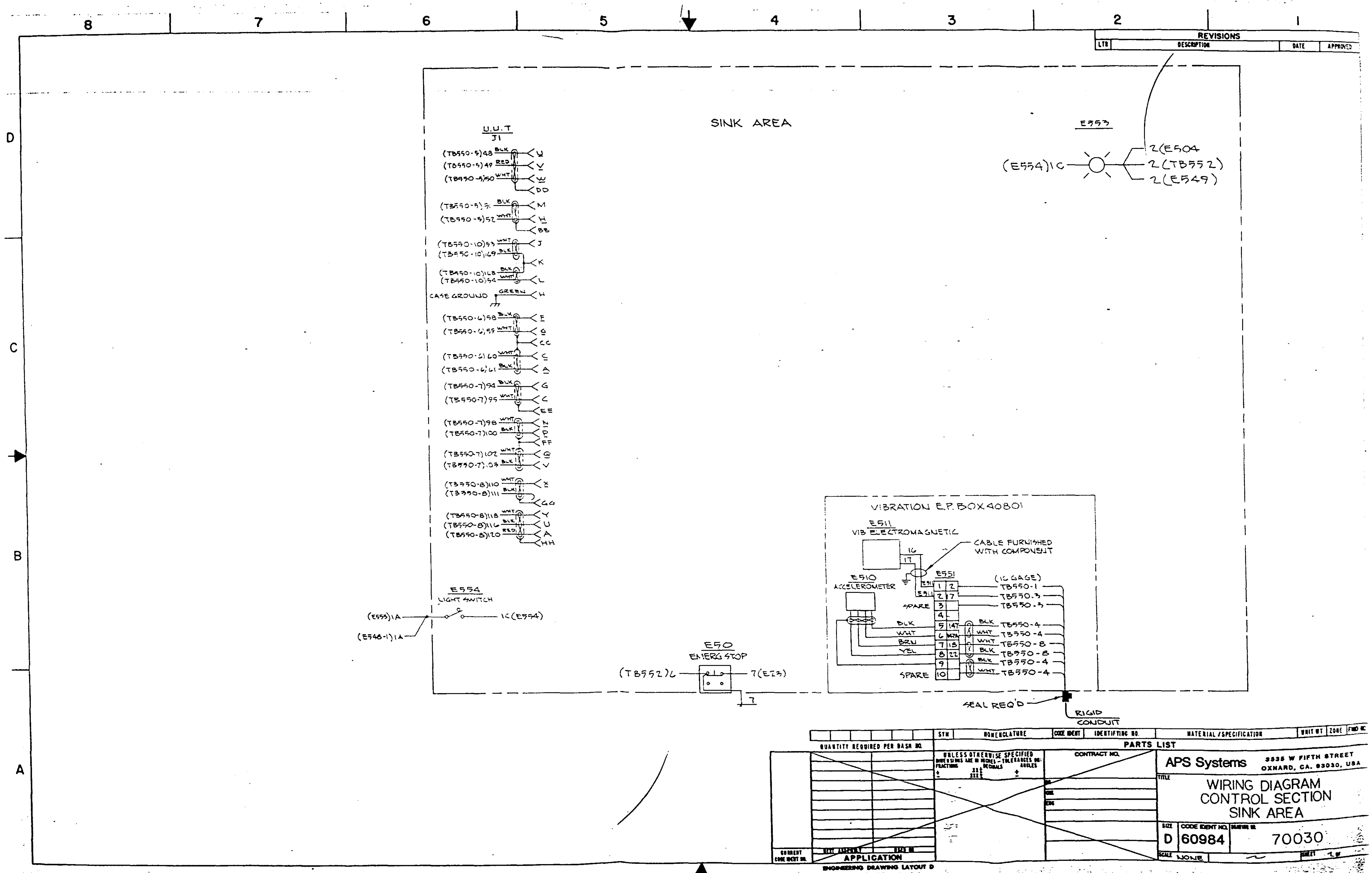


Figure II-4. Wiring Diagram Test Stand
(Sheet 2 of 3)

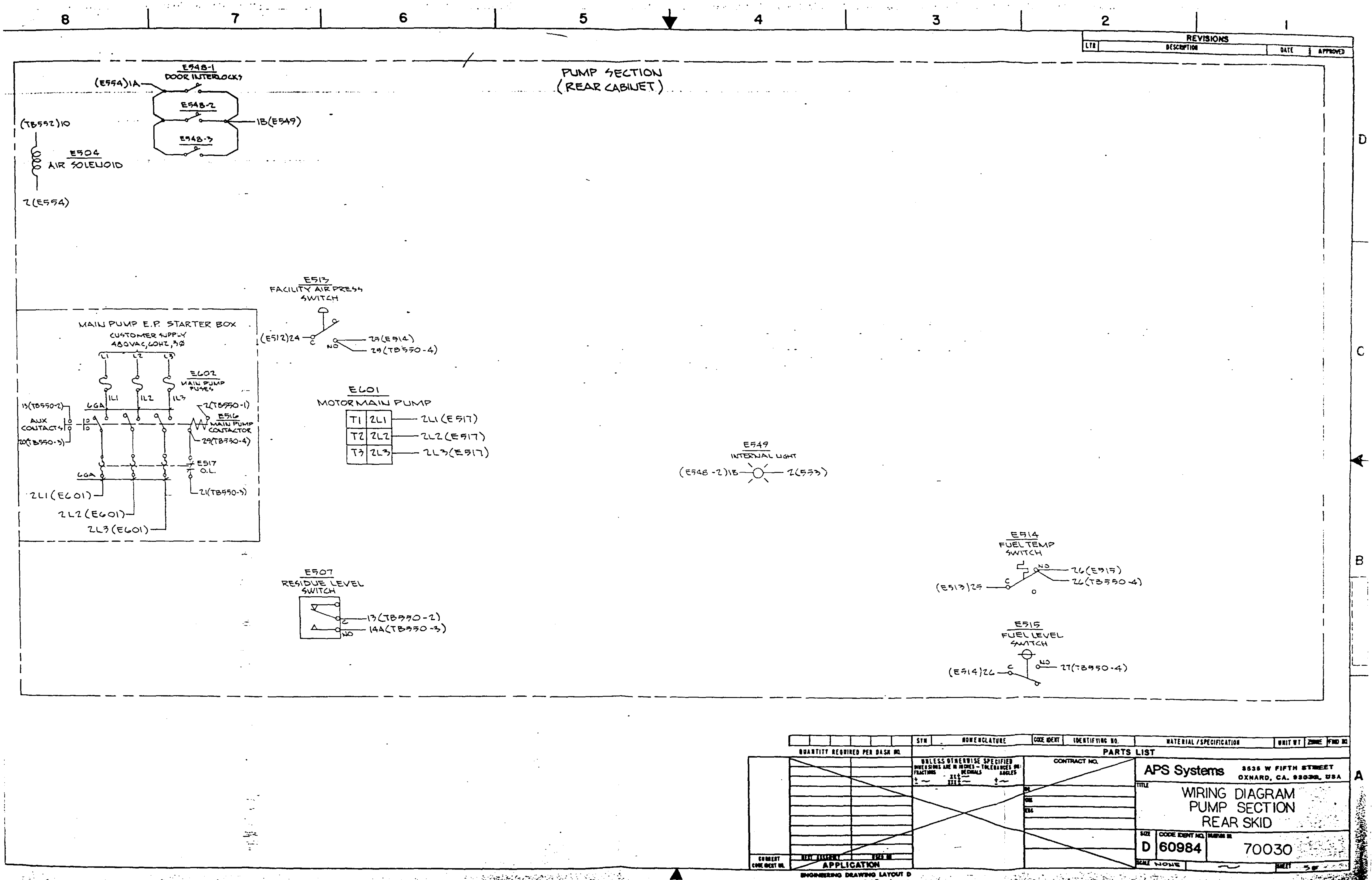


Figure II-4. Wiring Diagram Test Stand
(Sheet 3 of 3)

CHAPTER XII

PARTS LIST

12-1 GENERAL

This chapter contains the parts list of the major components of the Rate of Flow Test Stand Model 10047-10 (PL 10047 (9 sheets) and PL 20071 (17 sheets)). The item numbers on the parts list correspond to the numbers on the engineering drawings and schematics.

NOTE

FSCM is the Federal Supply Code for Manufacturers. The names and addresses of the manufacturers identified are provided in Table 12-1 before the parts list.

Table 12-1. FSCM List

FSCM NO.	NAME	ADDRESS
0AYU9	Industrial Computer Source ICS Computer Products Inc.	4837 Mercury Street San Diego, CA 92111
0CY71	C & K Components Inc.	2035 Highway 70E Clayton, NC 27520-0687
0DU12	Altech Corporation	349 Route 206 South Sommerville, NJ 08876
0EJN5	Elsie Manufacturing Co.	600 West Maple Street Waterloo, IN 46793
00779	AMP Incorporated	2800 Fulling Mill P. O. Box 3608 Harrisburg, PA 17105
02654	Daytronic Corporation	2589 Corporate Place Miamisburg, OH 45342-3655
02660	Amphenol Corporation Industrial Technology Division	358 Hall Avenue P. O. Box 384 Wallingford, CT 06492
04579	Aurora Pump A Unit of General Signal Corporation	800 Airport Road North Aurora, IL 60542-1403
04713	Motorola Incorporated Semiconductor Products Sector	5005 East McDowell Road Phoenix, AZ 85008-4229
04845	Automatic Switch Co.	50-60 Hanover Road Florham Park, NJ 07932
05840	Foxboro/Jordan	5607 West Douglas Avenue Milwaukee, WI 53218-1613
07282	Fairchild Industries Incorporated Industrial Products Div.	1501 Fairchild Drive Winston-Salem, NC 27105-4551
07342	North Atlantic Industries Inc.	60 Plant Avenue Hauppauge, NY 11788-3890
08004	Todd Systems Inc.	50 Ash Street Yonkers, NY 10701-3906

Table 12-1. FSCM List - Continued

FSCM NO.	NAME	ADDRESS
09393	Rochester Gauges Inc.	11616 Harry Hines Blvd. P. O. Box 29242 Dallas, TX 75229
12697	Clarostat Mfg. Co., Inc.	Lower Washington Street Dover, NH 03820
13169	Cleveland Vibrator Co.	2828 Clinton Avenue Cleveland, OH 44113-2939
13919	Burr-Brown Corporation	6730 South Tucson Blvd. P. O. Box 11400 Tucson, AZ 85734
16067	California Instruments	5150 Convoy Street San Diego, CA 92111-1208
16327	Dayton Electric Mfg. Co.	5959 West Howard Street Chicago, IL 60648-4014
16626	Contromatics Inc.	222 Roberts Street East Hartford, CT 06108
16951	Quantum Dynamics Co.Inc.	6414 Independence Avenue Woodland Hills,CA 91367-2685
22508	Fisher Controls Co. Inc.	205 South Center Street Marshalltown, IA 50158-2823
24123	Gast Manufacturing Corp.	2300 Highway M-139 P. O. Box 97 Benton Harbor, MI 49022-3561
24346	Lube Devices, Inc.	1864 Nagle Avenue P. O. Box 1148 Manitowoc, WI 54220-1702
24522	Humphrey Products Co.	5070 East North Avenue P. O. Box 2008 Kalamazoo, MI 49003
26337	Figgie Int'l Corp. Greer Hydraulics Div.	1340 E. 222nd Street Euclid, OH 44117
27385	Viatran Corporation	300 Industrial Drive Grand Island, NY 14072-1265
29907	Omega Engineering Inc.	One Omega Drive P. O. Box 4047 Stamford, CT 06907-0047

Table 12-1. FSCM List - Continued

FSCM NO.	NAME	ADDRESS
32873	Newport Electronics	2229 South Yale Street Santa Ana, CA 92704-4426
35705	Crouse-Hinds Canada Ltd.	1111 Elias Street P. O. Box 5693 Terminal A London Ont, Canada N6A4R9
38151	Marathon Electric Mfg. Corporation	100 East Randolph Street Wausau, WI 54401-2561
39428	McMaster-Carr Supply Co.	600 County Line Road Elmhurst, IL 60126
43990	Norgren C.A. Company	5400 South Delaware Street Littleton, CO 80120-1663
44655	Ohmite Manufacturing Co.	3601 West Howard Street Skokie, IL 60076-4014
49234	Protectoseal Company	225 West Foster Avenue Bensenville, IL 60106-1631
50139	Allen-Bradley Company Electronics Division	1414 Allen Bradley Drive El Paso, TX 79936-6415
50443	Hewlett-Packard Company	5301 Stevens Creek Blvd. Santa Clara, CA 95050
53111	Dean Products, Inc.	985 Dean Street Brooklyn, NY 11238-3304
53153	Lin-Act Mfg. Company	P. O. Box 2062 Rockford, IL 61111
53367	Setra Systems Inc.	45 Nagog Park Acton, MA 01720-3413
53651	Action Instruments	8601 Aero Drive San Diego, CA 92123-1724
54407	Power-One DC Power Supplies	740 Calle Plano Drive Camarillo, CA 93010-8555
55442	OPTO 22	15461 Springdale Street Huntington Beach, CA 92649-1335
55841	Cutler-Hammer, Div. of Eaton Corp. Electrical Construction Equipment Operations	4201 North 27th Street Milwaukee, WI 53216-1807

Table 12-1. FSCM List - Continued

FSCM NO.	NAME	ADDRESS
57797	Appleton Electric Co.	1701 West Wellington Ave. Chicago, IL 60657-4027
59793	Hycon Corporation	City Line Rd. & Ave. C. Bethlehem, PA 18017
60984	APS Systems	3535 West Fifth Street Oxnard, CA 93030-6498
60984	SATE c/o APS Systems	3535 West Fifth Street Oxnard, CA 93030-6498
61485	Hitachi Denshi America Limited	175 Crossways Park West Woodbury, NY 11797-2002
63590	SPC Technology	4801 North Ravenswood Ave. Chicago, IL 60640-4496
64667	National Instruments Corporation	6504 Bridge Point Parkway Austin, TX 78730-5039
65035	Rexroth Corporation Pneumatics Division	1953 Mercer Road Lexington, KY 40511
71400	Bussmann Division of Cooper Industries Inc.	114 Old State Road P. O. Box 14460 St. Louis, MO 63178
72433	Flexitallic Gasket Company Inc.	8440 Remington Avenue P. O. Box 286 Pennsauken, NJ 08110
73389	Adalet-PLM Division Scott and Fetzer Co.	4801 West 150th Street Cleveland, OH 44135-3301
74400	Hobbs Division Stewart-Warner Corp.	Yale Blvd. & Ash Street Springfield, IL 62705
75665	Lovejoy Incorporated	2655 Wisconsin Avenue Downers Grove, IL 60515-4229
77342	Potter & Brumfield Inc. A Siemens Company	200 S. Richland Creek Dr. Princeton, IN 47671-0001
83008	Staco Energy Products Company	301 Gaddis Boulevard Dayton, OH 45403-1314
83330	Dialight Corporation	1913 Atlantic Avenue Manasquan, NJ 08736-1005

Table 12-1. FSCM List - Continued

FSCM NO.	NAME	ADDRESS
85274	Dwyer Instruments Inc.	Jct 212 and US12 P. O. Box 373 Michigan City, IN 46360
86184	CLA-VAL Company	1701 Placentia Street Costa Mesa, CA 92627-4475
86768	Teledyne Republic Mfg. Headquarters Div. of Teledyne Inc.	15655 Brookpark Road Cleveland, OH 44142-1619
88078	Cashco Incorporated	Highway 140 P. O. Box 6 Ellsworth, KS 67439-0006
89020	Amerace Corporation Industrial Electrical Products	530 West Mt. Pleasant Ave. Livingston, NJ 07039
89395	Anderson W.E. Division of Dwyer Instruments Inc.	320 West 80th Street Kansas City, MO 64114-2315
9C599	Mallory Distributor Products Emhart Electrical/ Electronic Group	3029 East Washington Street Indianapolis, IN 46206
92021	PBM, Incorporated	RD6, Box 387-A Sandy Hill Road Irwin, PA 15642-9806
95146	Alco Electronic Products Incorporated	1551 Osgood Street North Andover, MA 01845-1014
98235	Kates WA Company	1450 Jarvis Avenue Ferndale, MI 48220
98991	Worcester Controls Corporation	33 Locke Drive P. O. Box 538 Marlborough, MA 01752-9906
	Matco Products, Inc.	P. O. Box 27 Route 22 Brewster, NY 10509
	Transcomputer, Inc.	1257 Tasman Drive Sunnyvale, CA 94089

PARTS LIST	DESIGN ACTIVITY: APS SYSTEMS 3535 WEST FIFTH STREET OXNARD, CA 93030-6498	CODE IDENT. 60984	PL 10047 / SHEET 1 OF 9 SHEETS	CONTRACT NO. F41608-88-C-2452
LIST TITLE	TEST STAND, RATE OF FLOW		REVISION LETTER/DATE: 04/30	SPECIFICATION PD-T-84-005A / AUTHENTICATION / REV / REV / REV

ITEM NO.	DWG. NO.	QTYREQD	QTYREQD -10	SYM	NOMENCLATURE	IDENTIFYING NO.	MATERIAL/SPECIFICATION	FSCM	SMR CODE
500	10047		1		RESERVOIR, MAIN	40813-10	450 GAL.CAPACITY, 304S.S.	60984	XB
501			1		RESERVOIR, WASTE	40867-10	10 GAL. CAPACITY, 304S.S.	60984	XB
502			1		VALVE, SHUT-OFF	5024K23	(PUMP INLET) 6" WAFER STYLE WITH LEVER-LOCK HANDLE, BUTTERFLY, BUNA-"N" SEAT, DUCTILE IRON BODY, DUCTILE IRON DISC	39428	PAFZZ
503			1		NOT USED				
504			1		PUMP, MAIN	344A-BF-2-1/2 X 3 X 9	CENTRIFUGAL, FOOT MOUNTING, HORIZONTAL, WITH POSITION (4) DISCHARGE FLANGE. CAPACITY 300 GPM @ 110PSIG DISCHARGE PRESSURE, 3,500 RPM,STANDARD FITTED PARTS, MECHANICAL SEAL,SERVICE MIL-C-7024B TYPE II CALIBRATION FLUID, SP.GR. 0.77 & VISCOSITY 1.15 C.S. @ 80 DEG F OPERATING TEMP. IMPELLER TRIMMED TO SUIT ABOVE OPERATING CONDITIONS.	04579	AFF
505			1		VALVE, CHECK	MODEL-501 P/N: 9995103A	(PUMP INLET) 3"-150LBS, WAFER SWING WITH VITON SEAT	86184	PAFZZ
506	V		3		EXCHANGER, HEAT	22" X 47"-S-2-B-302-14/14/CSTL	PLATE TYPE, CARBON STL,	53111	PAFZZ

PARTS LIST	DESIGN ACTIVITY: APS SYSTEMS 3535 WEST FIFTH STREET OXNARD, CA 93030-6498	CODE IDENT. 60984	PL 10047	CONTRACT NO. F41608-88-C-2452
			SHEET 2 OF 9 SHEETS	SPECIFICATION PD-T-84-005A
LIST TITLE	TEST STAND, RATE OF FLOW		REVISION LETTER/DATE 04/30	AUTHENTICATION REV REV REV

ITEM NO.	DWG. NO.	QTYREQD	QTYREQD -10	SYM	NOMENCLATURE	IDENTIFYING NO.	MATERIAL/SPECIFICATION	FSCM	SMR CODE
507	10047		1		VALVE, FLOW	1-1/2"-513-GS	1-1/2", SCREWED ENDS, BRONZE BODY	22508	AFF
508			1		FILTER, MAIN	RFLP1300F10B	300GPM OF MIL-C-7024B TYPE II CALIBRATION FLUID, 10 MICRON ELEMENTS	59793	PBFFF
509			1		ACCUMULATOR	800761	5 GAL CAPACITY, COMPATIBLE W/ MIL-C-7024B TYPE II FLUID, PRECHARGE TO 50PSIG NITROGEN, W/MOUNTING BRACKETS RUBBER RING & CLAMP	26337	XB
509-1			1		BRACKET	ZMB-89		26337	XB
509-2			1		BASE, BRACKET	Z-67R		26337	XB
510			1		VALVE, RELIEF	63EG-6365	(SYSTEM SAFETY) 2", SCREWED ENDS, BUNA-"N" SOFTWARE, 35-125PSIG SPRING RANGE, MIL-C-7024B CALIBRATION FLUID SERVICE, SET AT 85PSIG	22508	AFF
511			1		VALVE, SHUT-OFF	1/2" MODEL 200SR-C-11-11-BC-FP	(LOW FLOW)1/2"NPT,AIR OPERATED BALL VALVE,FULL PORT,CV=8.5,NC	16626	PBFFF
512	V		1		VALVE, SHUT-OFF	3/4" MODEL 200SR-B-11-11-BC-FP	(MEDIUM FLOW)3/4"NPT,AIR OPERATED BALL VALVE, FULL PORT, CV=16.5, NC	16626	PBFFF

PARTS LIST	DESIGN ACTIVITY: APS SYSTEMS 3535 WEST FIFTH STREET OXNARD, CA 93030-6498	CODE IDENT. 60984	PL 10047	CONTRACT NO. F41608-88-C-2452
			SHEET 3 OF 9 SHEETS	SPECIFICATION PD-T-84-005A
LIST TITLE	TEST STAND, RATE OF FLOW		REVISION LETTER/DATE 04/30	AUTHENTICATION / REV / REV / REV

ITEM NO.	DWG. NO.	QTYREQD	QTYREQD -10	SYM	NOMENCLATURE	IDENTIFYING NO.	MATERIAL/SPECIFICATION	FSCM	SMR CODE
513	10047		1		VALVE, SHUT-OFF	3" MODEL 375SR-C-11-11-BB	(HIGH FLOW) 3"-150LBS, AIR OPERATED BALL VALVE, STEEL BODY & BALL, CV=240, NC	16626	PBFFF
514			1		CONTROLLER, FLOW RATE	CB11T-H	(LOW FLOW) 1/2"NPT ENDS, PNEU-MATIC ACTUATOR FOR 3-15PSI CONTROL SIGNAL, FLUID MEDIA MIL-C-7024B CALIBRATION FLUID, INLET PRESSURE 130PSIG	98235	AFF
515			1		CONTROLLER, FLOW RATE	GB11T-H	(MEDIUM FLOW) 3/4"NPT ENDS, PNEUMATIC ACTUATOR FOR 3-15PSI CONTROL SIGNAL, FLUID CRITERIA -SAME AS ITEM # 514	98235	AFF
516			1		CONTROLLER, FLOW RATE	MA11B-H	(HIGH FLOW) 3"-150LBS RF FLANGES, PNEUMATIC ACTUATOR FOR 3-15PSI CONTROL SIGNAL, FLUID CRITERIA-SAME AS ITEM # 514	98235	AFF
517			1		VALVE, DIVERTER	DPB19F152P0C1	(FLOW BYPASS) 3"-150LBS, FLAN-GED, 3-WAY, BRONZE BODY, SIDE ENTRY, ANGLE PORT BALL & #10 FLOW PATTERN	92021	PBFFF
517-1	V		1		ACTUATOR	CB520SR80	SPECIAL INCREASED PORTS ON PNEUMATIC ACTUATOR TO ALLOW 1/2 SECOND ACTUATION TIME	92021	PBFFF

PARTS LIST	DESIGN ACTIVITY: APS SYSTEMS 3535 WEST FIFTH STREET OXNARD, CA 93030-6498	CODE IDENT. 60984	PL 10047	CONTRACT NO. F41608-88-C-2452
			SHEET 4 OF 9 SHEETS	SPECIFICATION PD-T-84-005A
LIST TITLE TEST STAND, RATE OF FLOW			REVISION LETTER/DATE 04/30	AUTHENTICATION / REV / REV / REV

ITEM NO.	DWG. NO.	QTYREQD	-10	SYM	NOMENCLATURE	IDENTIFYING NO.	MATERIAL/SPECIFICATION	FSCM	SMR CODE
518	10047		1		VALVE, SHUT-OFF	2" MODEL 275SR-C-11-11-BC-FP	(FLOWMETER SELECT) 2"NPT, AIR OPERATED BALL VALVE, FULL PORT, CV=195, NO	16626	PBFFF
519			1		REGULATOR, PRESSURE	1"657-EZ-67AF-3582	(BACK PRESSURE) 1"NPT, STEEL BODY WITH POSITIONER, MICRO-FLAT PLUG,3-9.5PSI SPLIT RANGE	22508	AFF
520			1		REGULATOR, PRESSURE	2"-1052-V100-67AF-3610J	(BACK PRESSURE) 2"NPT, STEEL BODY WITH POSITIONER, VEE BALL PLUG, 9-15PSI SPLIT RANGE	22508	AFF
521			1		FILTER/REGULATOR WITH GAGE	B12-697-M1LA	(FACILITY AIR) 1/2"NPT PORTS, 0-160PSIG, 5 MICRON	43990	XB
521-1			1		ELEMENT	2992-18		43990	PAFZZ
522			1		FILTER, COALESCING	F45-621-MODA	(FACILITY AIR) 1/2"NPT PORTS	43990	XB
522-1			1		MOUNTING BRACKET	5532-04		43990	PAFZZ
522-2			1		ELEMENT	5350-08		43990	PAFZZ
523			1		VALVE, SHUT-OFF	1"-444TSE	(RESERVOIR FILL) BALL VALVE-1"NPT	98991	PAFZZ
524	V		1		FILTER	LFBH/HC110G10B1.0/-5	(PRESSURIZED FILL) MIL-C-7024B TYPE II CALIBRATION FLUID, 10 MICRON	59793	PAFZZ

PARTS LIST	DESIGN ACTIVITY: APS SYSTEMS 3535 WEST FIFTH STREET OXNARD, CA 93030-6498	CODE IDENT. 60984	PL 10047	CONTRACT NO. F41608-88-C-2452
			SHEET 5 OF 9 SHEETS	SPECIFICATION PD-T-84-005A
LIST TITLE	TEST STAND, RATE OF FLOW		REVISION LETTER/DATE 04/30	AUTHENTICATION REV REV REV

ITEM NO.	DWG. NO.	QTYREQD	QTYREQD -10	SYM	NOMENCLATURE	IDENTIFYING NO.	MATERIAL/SPECIFICATION	FSCM	SMR CODE
525	10047		1		CAP, SAFETY FILL	1274	(RESERVOIR FILL) 2-1/2" NPT FEMALE FITTING, 6-3/4" FLAME ARRESTOR LENGTH.	49234	PAFZZ
526			1		VENT, FLAME ARRESTOR	4951	1" NPT, ALUMINUM	49234	XB
527			1		PUMP, RESIDUE	1P796A	(SINK DRAIN, TRANSFER) CENTRI-FUGAL, BRONZE BODY	16327	PAFZZ
528			1		MOTOR, AIR	2AM-FCW-13 WITH AH105L LUBRICATOR	(RESIDUE PUMP)	24123	XB
529			1		REGULATOR, PRESSURE (WITH GAGE)	R08-200-RGMA	(PURGE PRESSURE) 0-160PSIG RANGE, SET @ 10PSIG	43990	PAFZZ
530			1		VALVE, SHUT-OFF	730T08	(EMERGENCY DRAIN) 2"NPT, GATE TYPE, BRONZE BODY	MATCO	PAFZZ
531			1		VALVE, SHUT-OFF	4726K14	(WATER INLET) 1-1/2"NPT, BALL VALVE, BRONZE	39428	PAFZZ
532			2		LINK, FUSIBLE	MODEL C - 50 LBS. 160 DEG F		0EJN5	PAFZZ
533			1		VALVE, CHECK	RV-12-1.0/12	(RESIDUE TRANSFER) STEEL BODY	59793	PAFZZ
534	V		1		GLASS, SIGHT	G620-5-A-1	ALUMINUM BODY, BUNA-N SEALS, KG-33 REDLINE GAGE CLASS, 1-3/8" ST.STL DIAL THERMOMETER	24346	PAFZZ

PARTS LIST	DESIGN ACTIVITY: APS SYSTEMS 3535 WEST FIFTH STREET OXNARD, CA 93030-6498	CODE IDENT. 60984	PL 10047	CONTRACT NO. F41608-88-C-2452
			SHEET 6 OF 9 SHEETS	SPECIFICATION PD-T-84-005A
LIST TITLE	TEST STAND, RATE OF FLOW		REVISION LETTER/DATE: 04/30	AUTHENTICATION / REV / REV / REV

ITEM NO.	DWG. NO.	QTYREQD	QTYREQD -10	SYM	NOMENCLATURE	IDENTIFYING NO.	MATERIAL/SPECIFICATION	FSCM	SMR CODE
535	10047		3		VALVE, SHUT-OFF	148-4FLDBP	(TRANSDUCER/SAMPLING) 1/4" MS PORTS	86768	XB
536			1		VALVE, DUMP	DP3-22M7-2E0	(HIGH BACK PRESSURE) 1/4"NPT PORTS, DEMI PACKLESS	88078	PAFZZ
537			1		VALVE, QUICK EXHAUST	P52935-6	(DIVERTER VALVE) 3/4"NPT PORTS, ALUMINUM BODY	65035	PAFZZ
538			1		VALVE, CHECK	MODEL-501 P/N: 999-5102C	(RETURN CHECK)2-1/2"-150LB, WAFER SWING WITH VITON SEAT	86184	PAFZZ
539			1		GAGE, LEVEL	MODEL 6560	(RESERVOIR)FLOAT LENGTH 21.5", TUBE LENGTH 25" WITH 22-05246 ADAPTER	09393	PAFZZ
540			1		AMPLIFIER, AIR	5571K12	(PURGE) EDUCTOR .15"W.C.	39428	PAFZZ
541			1		CYLINDER, CLAMPING	STTC-2X4-4	(UUT CLAMPING) AIR OPERATED	53153	PAFZZ
541-1			1		CLEVIS, ROD END	L-97-4		53153	PAFZZ
542			1		VALVE, 4-WAY	MODEL 41V	(CYLINDER ACTUATING)	24522	PAFZZ
543			1		REGULATOR, PRESSURE (WITH GAGE)	R08-200-RGMA	(INSTRUMENT AIR) 0-160PSIG RANGE, SET @ 20PSIG	43990	PAFZZ
544	V		1		CABINET, FRONT	40877-10	(CONTROL SECTION) 14GA STL CONSTRUCTION	60984	MFF

PARTS LIST	DESIGN ACTIVITY: APS SYSTEMS 3535 WEST FIFTH STREET OXNARD, CA 93030-6498	CODE IDENT. 60984	PL 10047 SHEET 7 OF 9 SHEETS	CONTRACT NO. F41608-88-C-2452 SPECIFICATION PD-T-84-005A
LIST TITLE TEST STAND, RATE OF FLOW			REVISION LETTER/DATE 04/30	AUTHENTICATION | REV | REV | REV

ITEM NO.	DWG. NO.	QTYREQD	QTYREQD -10	SYM	NOMENCLATURE	IDENTIFYING NO.	MATERIAL/SPECIFICATION	FSCM	SMR CODE
545	10047		1		CABINET, REAR	40876-10	(PUMP SECTION) 14GA STL CONSTRUCTION	60984	MFF
546			1		FRAME, LOWER, REAR	40861-10	6" CHANNEL CONSTRUCTION	60984	MFF
547			1		FRAME, LOWER, FRONT	40871-10	6" CHANNEL CONSTRUCTION	60984	MFF
548			1		FRAME, LOWER, ELEC- TRICAL CABINET	40869-10	6" CHANNEL CONSTRUCTION	60984	MFF
549			1		CABINET, ELECTRICAL	40913-10	(HOFMANN C-1448BD CONSOLE)	60984	MFF
550			1		J-BOX, ELECTRICAL	40803-01	(START/STOP) X-PROOF	60984	XB
551			1		COUPLING, FLEXIBLE	L-070 WITH ONE BORE 0.500 & 1/8 KEY, OTHER BORE 0.625 & 3/16 KEY	(RESIDUE PUMP) JAW TYPE WITH SOX SPIDER	75665	PAFZZ
552			1		COUPLING, FLEXIBLE	L-110 WITH ONE BORE 1.625 & 3/8 KEY, OTHER BORE 1.125 & 1/4 KEY	(MAIN PUMP) JAW TYPE WITH SOX SPIDER	75665	XB
553			1		VALVE, RELIEF	40981-10	(PURGE) 0.15" W.C.	60984	PAFZZ
554	V				NOT USED				

PARTS LIST	DESIGN ACTIVITY: APS SYSTEMS 3535 WEST FIFTH STREET OXNARD, CA 93030-6498	CODE IDENT. 60984	PL 10047 SHEET 8 OF 9 SHEETS	CONTRACT NO. F41608-88-C-2452 SPECIFICATION PD-T-84-005A
LIST TITLE	TEST STAND, RATE OF FLOW		REVISION LETTER/DATE 04/30	AUTHENTICATION | REV | REV | REV

ITEM NO.	DWG. NO.	QTYREQD	QTYREQD -10	SYM	NOMENCLATURE	IDENTIFYING NO.	MATERIAL/SPECIFICATION	FSCM	SMR CODE
555	10047		1		J-BOX, ELECTRICAL	40802-01	(PRESSURE TRANSDUCER) X-PROOF	60984	XB
556			60		BOLT, HEX HEAD	91247A805	GRADE 5, ZINC PLATED	39428	PAFZZ
557			84		NUT, HEX	95462A035	GRADE 5, ZINC PLATED	39428	PAFZZ
558			10		GASKET	3"-150LB ANSI	FLEXITALLIC, NON-ASBESTOS	72433	PAFZZ
559			4		BOLT, HEX HEAD	91247A822	GRADE 5, ZINC PLATED	39428	PAFZZ
560			6		GASKET	2"-150LB ANSI	FLEXITALLIC, NON-ASBESTOS	72433	PAFZZ
561			8		BOLT, HEX HEAD	91247A814	GRADE 5, ZINC PLATED	39428	PAFZZ
562			2		GASKET	2-1/2"-150LB ANSI	FLEXITALLIC, NON-ASBESTOS	72433	PAFZZ
563			2		GASKET	4"-150LB ANSI	FLEXITALLIC, NON-ASBESTOS	72433	PAFZZ
564			1		GASKET	40992-01	FLAT FACE, .06"THK BUNA-N	60984	PAFZZ
565			12		BOLT, HEX HEAD	91247A800	GRADE 5, ZINC PLATED	39428	PAFZZ
566			1		PANEL ASSY., INSTR.	40870-10	(UPPER PANEL)	60984	AFF
566-1			1		PANEL	40873-01		60984	MFF
567			1		PANEL ASSY.,CONTROL	40872-10	(LOWER PANEL)	60984	AFF
567-1	V		1		PANEL	40874-01		60984	MFF

PARTS LIST	DESIGN ACTIVITY: APS SYSTEMS 3535 WEST FIFTH STREET OXNARD, CA 93030-6498	CODE IDENT. 60984	PL 10047 SHEET 9 OF 9 SHEETS	CONTRACT NO. F41608-88-C-2452 SPECIFICATION PD-T-84-005A
LIST TITLE	TEST STAND, RATE OF FLOW		REVISION LETTER/DATE: 04/30	AUTHENTICATION / REV / REV / REV

ITEM NO.	DWG. NO.	QTYREQD	-10	SYM	NOMENCLATURE	IDENTIFYING NO.	MATERIAL/SPECIFICATION	FSCM	SMR CODE
568	10047		1		J-BOX, ELECTRICAL	40801-01	(VIBRATOR) X-PROOF	60984	PBFFF
569			1		STARTER BOX ASSY MAIN MOTOR/PUMP	20072-10		60984	AFF
570 to 579					NOT USED				
580	V		1		ELECTRICAL CONT. PANEL ASSY	20071-10		60984	AFF

PARTS LIST	DESIGN ACTIVITY: APS SYSTEMS 3535 WEST FIFTH STREET OXNARD, CA 93030-6498	CODE IDENT. 60984	PL 20071	CONTRACT NO. F41608-88-C-2452
			SHEET 1 OF 17 SHEETS	SPECIFICATION PD-T-84-005A
LIST TITLE	TEST STAND, RATE OF FLOW		REVISION LETTER/DATE 04/30	AUTHENTICATION / REV / REV / REV

ITEM NO.	DWG. NO.	QTYREQD	-10	SYM	NOMENCLATURE	IDENTIFYING NO.	MATERIAL/SPECIFICATION	FSCM	SMR CODE
E1	20071		1		LIGHT, INDICATOR	XLSRS	(PURGE FAIL) 120VAC, RED, X-PROOF	73389	PAFZZ
E2			1		PUSHBUTTON W/LIGHT	XLPGS	(SYSTEM START)120VAC,GRN, 1N.O.,1N.C. CONTACT, X-PROOF	73389	PAFZZ
E2-1			1		BLOCK, CONTACT	BTIA		73389	PAFZZ
E3			1		PUSHBUTTON	XHPBS	(SYSTEM STOP)120VAC,RED,1N.O., 1N.C. CONTACT, X-PROOF	73389	PAFZZ
E3-1			1		BLOCK, CONTACT	BTIA		73389	PAFZZ
E4			1		VOLTMETER	Q2010FVR5-D1	(400HZ VOLTS) 0-199.9 VTRMS WITH D1 CONNECTOR	32873	PAFFF
E5			1		AMMETER	Q2010GCR6-D1	(400HZ AMPS) 0-1.999 ATRMS WITH D1 CONNECTOR	32873	PAFFF
E6			1		VOLTMETER	Q2010FVR5-D1	(LOW FREQ. VOLTS) 0-199.9 VTRMS WITH D1 CONNECTOR	32873	PAFFF
E7			1		AMMETER	Q2010GCR6-D1	(LOW FREQ. AMPS) 0-1.999 ATRMS WITH D1 CONNECTOR	32873	PAFFF
E8	V		1		POWER SUPPLY	LF6-35/15-04.0-3/70-06.0-2/28-06.6-2/60-08.0-2/SPARE/SPARE	(LOW FREQUENCY)0-75VAC,4-10HZ, SQ. WAVE, 2/3 PHASE ADJUSTABLE	32873	PAFDD

PARTS LIST	DESIGN ACTIVITY: APS SYSTEMS 3535 WEST FIFTH STREET OXNARD, CA 93030-6498	CODE IDENT. 60984	PL 20071	CONTRACT NO. F41608-88-C-2452
			SHEET 2 OF 17 SHEETS	SPECIFICATION PD-T-84-005A
LIST TITLE TEST STAND, RATE OF FLOW			REVISION LETTER/DATE: 04/30	AUTHENTICATION / REV / REV / REV

ITEM NO.	DWG. NO.	QTYREQD	QTYREQD -10	SYM	NOMENCLATURE	IDENTIFYING NO.	MATERIAL/SPECIFICATION	FSCM	SMR CODE
E9	20071		1		TRANSFORMER, VARIABLE	291	(VIBRATION AMPLITUDE ADJUST) 0-120VAC, 3AMP.	83008	PAFZZ
E10			1		SWITCH, SELECTOR	10250T20KB	(VIBRATION ON) 2 POSITION MAINTAINED CONTACT 120VAC	55841	PAFZZ
E11			1		LIGHT, INDICATOR	125-1308-11-143	(VIBRATION ON) 120VAC, GREEN	83330	PAFZZ
E11-1			1		CAP	125-1131-403(GREEN)		83330	PAFZZ
E11-2			1		LAMP	B1A(NE-51)		83330	PAFZZ
E12			1		VOLTMETER	Q2000CVR3	(VIBRATION 'MILLS') 0-1.999 VAC SCALED TO READ IN MILLS.	32873	PAFFF
E12-1			1		POTENTIOMETER	RV4LAYSA103A	(VIBRATION SCALING) 0-10KOHMS	12697	PAFZZ
E13			1		INDICATOR, PRESSURE	MODEL 3270	(INLET PRESSURE) DIGITAL READ-OUT WITH 80% SHUNT CAL. AND BRIDGE EXCITATION	02654	PAFFF
E14			1		HOURMETER	20017-14	(PUMP HOURS) 120VAC, 0-99999.9 HRS	74400	PAFZZ
E15	V		1		CONTROLLER, TEMPERATURE	CN383-TC-A	(FUEL TEMPERATURE) MICROPRO-CESSOR BASED FOR TYPE 'T' THERMOCOUPLE INPUT, 4-20mA OUTPUT, DUAL ALARM CONTACT OUTPUTS (2.5A)	29907	PAFFF

PARTS LIST	DESIGN ACTIVITY: APS SYSTEMS 3535 WEST FIFTH STREET OXNARD, CA 93030-6498	CODE IDENT. 60984	PL 20071	CONTRACT NO. F41608-88-C-2452
			SHEET 3 OF 17 SHEETS	SPECIFICATION PD-T-84-005A
LIST TITLE TEST STAND, RATE OF FLOW			REVISION LETTER/DATE 04/30	AUTHENTICATION

ITEM NO.	DWG. NO.	QTYREQD	-10	SYM	NOMENCLATURE	IDENTIFYING NO.	MATERIAL/SPECIFICATION	FSCM	SMR CODE
E16	20071		1		PUSHBUTTON	10250T25R	(PUMP/UUT POWER OFF) 120VAC, RED, 1 N.C. CONTACT	55841	PAFZZ
E17			1		PUSHBUTTON	10250T23B	(PUMP/UUT POWER ON) 120VAC, BLACK, 1 N.O. CONTACT	55841	PAFZZ
E18			1		LIGHT, INDICATOR	10250T34A	(PUMP/UUT ON) 120VAC, AMBER	55841	PAFZZ
E18-1			1		LAMP	755		55841	PAFZZ
E19			1		INDICATOR, PRESSURE	MODEL 3270	(DIFFERENTIAL PRESSURE)DIGITAL READOUT WITH 80% SHUNT CAL. AND BRIDGE EXCITATION	02654	PAFFF
E20			1		INDICATOR, PRESSURE	MODEL 3270	(BACK PRESSURE) DIGITAL READ-OUT WITH 80% SHUNT CAL. AND BRIDGE EXCITATION	02654	PAFFF
E21			1		LIGHT, INDICATOR	125-1308-11-143	(SYSTEM FAULT) 120VAC, RED	83330	PAFZZ
E21-1			1		CAP	125-1131-403 (RED)		83330	PAFZZ
E21-2			1		LAMP	BIA(NE-51) LAMP		83330	PAFZZ
E22			1		ALARM, AUDIBLE	SC110N	(SYSTEM ALARM) 120VAC	9C599	PAFZZ
E23	V		1		PUSHBUTTON	XHPBMH	(EMERGENCY STOP) 120VAC, RED, MUSHROOM HEAD 1 N.O., 1 N.C. MOMENTARY CONTACT, X-PROOF	73389	PAFZZ

PARTS LIST	DESIGN ACTIVITY: APS SYSTEMS 3535 WEST FIFTH STREET OXNARD, CA 93030-6498	CODE IDENT. 60984	PL 20071	CONTRACT NO. F41608-88-C-2452
			SHEET 4 OF 17 SHEETS	SPECIFICATION PD-T-84-005A
LIST TITLE	TEST STAND, RATE OF FLOW		REVISION LETTER/DATE 04/30	AUTHENTICATION REV REV REV

ITEM NO.	DWG. NO.	QTYREQD	-10	SYM	NOMENCLATURE	IDENTIFYING NO.	MATERIAL/SPECIFICATION	FSCM	SMR CODE
E23-1	20071		1		BLOCK, CONTACT	BTIA		73389	PAFZZ
E23-2			1		HOUSING	XA3		73389	PAFZZ
E24 THRU E29					NOT USED				
E30			1		VOLTMETER	Q2000CVR4-D1	(LINEAR SERVO) 0-19.99VAC WITH D1 CONNECTOR	32873	PAFZZ
E31			1		INDICATOR, PHASE ANGLE	8810-F3210	(UUT DEGREES) 0-359.999 DEG +/- .001 DEG RESOLUTION	07342	PBFFF
E32			1		OSCILLOSCOPE	MODEL V-212	(TRANSIENT MONITOR) 20MHZ, DUAL CHANNEL	61485	PBFFF
E32-1			1		KIT, MOUNTING	RM-1065C		61485	PAFZZ
E33			1		JACK	31-4803BNC	(SCOPE INPUT) BULKHEAD CONNECTOR	02660	PAFZZ
E34			1		JACK	31-4803BNC	(REMOTE FLOW INPUT) BULKHEAD CONNECTOR	02660	PAFZZ
E35	V		1		SWITCH, ROTARY	A124-05-R-N-Z-B	(MODE SELECT)110VAC,350mA,SP4 POS.90 DEG INDEX	0CY71	PAFZZ

PARTS LIST	DESIGN ACTIVITY: APS SYSTEMS 3535 WEST FIFTH STREET OXNARD, CA 93030-6498	CODE IDENT. 60984	PL 20071	CONTRACT NO. F41608-88-C-2452
			SHEET 5 OF 17 SHEETS	SPECIFICATION PD-T-84-005A
LIST TITLE	TEST STAND, RATE OF FLOW		REVISION LETTER/DATE: 04/30	AUTHENTICATION / REV / REV / REV

NO.	DWG. NO.	QTYREQD	QTYREQD -10	SYM	NOMENCLATURE	IDENTIFYING NO.	MATERIAL/SPECIFICATION	FSCM	SMR CODE
E36	20071		1		SWITCH, ROTARY	A112-05-R-N-Z-B	(SCOPE INPUT SELECT) 110VAC, 350mA, SP12 POS. 30 DEG INDEX	0CY71	PAFZZ
E37			1		INDICATOR, FLOW, MASTER	LBL-406WR-SL-(M-TDC)	(MASTER FLOW) DIGITAL 50-120,000 PPH RANGE, TEMP/DENSITY COMPENSATED RS-232C PORT	16951	PBFDD
E38			1		OPERATOR TERMINAL #1	TM2500		13919	PBFFF
E39			1		SWITCH	164EL-11	(START FLOW) 20VDC, 0.4 VA, 1N.O.,1 N.C. MOMENTARY CONTACT	95146	PAFZZ
E40			1		SWITCH	164EL-11	(STOP FLOW/BYPASS) 20VDC,0.4VA 1 N.O., 1 N.C. MOMENTARY CONTACT	95146	PAFZZ
E41			1		SWITCH, KEY	164J2A-11	(PROGRAM/RUN) 20VDC, 0.4VA, 2 POSITION, 1 N.O., 1 N.C. MAINTAINED CONTACT	95146	PAFZZ
E42			1		OPERATOR TERMINAL #2	TM2500		13919	PBFFF
E43 THRU E49	V				NOT USED				

PARTS LIST	DESIGN ACTIVITY: APS SYSTEMS 3535 WEST FIFTH STREET OXNARD, CA 93030-6498	CODE IDENT. 60984	PL 20071	CONTRACT NO. F41608-88-C-2452
			SHEET 6 OF 17 SHEETS	SPECIFICATION PD-T-84-005A
LIST TITLE TEST STAND, RATE OF FLOW			REVISION LETTER/DATE 04/30	AUTHENTICATION / REV / REV / REV

ITEM NO.	DWG. NO.	QTYREQD	-10	SYM	NOMENCLATURE	IDENTIFYING NO.	MATERIAL/SPECIFICATION	FSCM	SMR CODE
E50	20071		1		PUSHBUTTON	XHPBMH	(EMERGENCY STOP) 120VAC, RED, MUSHROOM HEAD, 1 N.O., 1 N.C. MOMENTARY CONTACT X-PROOF	73389	PAFZZ
E50-1			1		BLOCK, CONTACT	BTIA		73389	PAFZZ
E50-2			1		HOUSING	XA3		73389	PAFZZ
E51 THRU E499					NOT USED				
E500			1		FUSE, 15AMP	MDA-15	(PURGE/INSTRUMENTATION) 300V, 15A, SLO-BLO, 1/4 X 1-1/4	71400	PAFZZ
E501			1		FUSE, 10AMP	MDA-10	(VENT MOTOR) 300V, 10A, SLO-BLO, 1/4 X 1-1/4	71400	PAFZZ
E501-1			2		FUSE HOLDER	S8300-3	3POLE, 300V, 1/4 X 1-1/4	71400	PAFZZ
E502			1		RELAY	CE15BN3A	(SYSTEM START/VENT FAN)115VAC, 15AMP, 3POLE	55841	PAFZZ
E503			1		SWITCH, DIFFERENT-IAL PRESSURE	1950-0	(PURGE PRESSURE) RANGE 0.15-0.5" H20, SPDT, 15A, 120VAC, X-PROOF	85274	PAFZZ
E504	V		1		VALVE, SOLENOID	8215C22	(FACILITY AIR) BRASS BODY 1/2 NPT, 120VAC, X-PROOF	04845	PAFZZ

PARTS LIST	DESIGN ACTIVITY: APS SYSTEMS 3535 WEST FIFTH STREET OXNARD, CA 93030-6498	CODE IDENT. 60984	PL 20071	CONTRACT NO. F41608-88-C-2452
			SHEET 7 OF 17 SHEETS	SPECIFICATION PD-T-84-005A
LIST TITLE	TEST STAND, RATE OF FLOW		REVISION LETTER/DATE: 04/30	AUTHENTICATION / REV / REV / REV

ITEM NO.	DWG. NO.	QTYREQD	QTYREQD -10	SYM	NOMENCLATURE	IDENTIFYING NO.	MATERIAL/SPECIFICATION	FSCM	SMR CODE
E505	20071		1		RELAY, TIMER	SSC12ANA	(PURGE TIMING) 120VAC, 1-30 MINS. ┬DPU	89020	PAFZZ
E505-1			1		SOCKET, RELAY	BCSA08SC	SCREW TERMINALS (FOR E505)	89020	PAFZZ
E506			1		VALVE, SOLENOID	8215C83	(FAST PURGE) ALUM. BODY, 2NPT, 120 VAC	04845	PAFZZ
E507			1		SWITCH, LEVEL	L4	(RESIDUE RESERVOIR) SPDT, 120VAC, 10A, X-PROOF	89395	PAFZZ
E508			1		VALVE, SOLENOID	8319C2	(RESIDUE TRANSFER PUMP) ALUMINUM BODY, 3/8NPT, 120VAC - BANK OF 6 EACH	04845	XB
E509			1		CONNECTOR, UUT	MS3114F22-55S	(UUT TEST)		PAFZZ
E509-1			1		DUST CAP	MS3181-22C3			PAFZZ
E510			1		ACCELEROMETER	141A	(UUT VIBRATION) +/- 2G RANGE CALIBRATED AT 12VDC EXCITATION	53367	XB
E511			1		VIBRATOR	CM-30	(TEST TABLE) ELECTROMAGNETIC, 60HZ	13169	XB
E512	V		1		SWITCH, PRESSURE	SB12BK/TE10A21	(FACILITY WATER) DPDT, 120VAC, 5A RANGE 2-60PSI SET @ 20PSI X-PROOF	04845	PAFZZ

PARTS LIST	DESIGN ACTIVITY: APS SYSTEMS 3535 WEST FIFTH STREET OXNARD, CA 93030-6498	CODE IDENT. 60984	PL 20071	CONTRACT NO. F41608-88-C-2452
			SHEET 8 OF 17 SHEETS	SPECIFICATION PD-T-84-005A
LIST TITLE TEST STAND, RATE OF FLOW			REVISION LETTER/DATE 04/30	AUTHENTICATION REV REV REV

ITEM NO.	DWG. NO.	QTYREQD	QTYREQD -10	SYM	NOMENCLATURE	IDENTIFYING NO.	MATERIAL/SPECIFICATION	FSCM	SMR CODE
E513	20071		1		SWITCH, PRESSURE	SB12BK/TF10A21	(FACILITY AIR) DPDT, 120VAC, 5A RANGE 3-100PSI SET @ 70PSI X-PROOF	04845	PAFZZ
E514			1		SWITCH, TEMPERATURE	SB12BK/QF10A1	(FUEL RESERVOIR) DPDT, 120VAC, 5A RANGE 50DEG-160DEG.F SET @ 90DEG +/- 2DEG.F, X-PROOF	04845	PAFZZ
E515			1		SWITCH, LEVEL	L4	(MAIN FUEL RESERVOIR) SPDT, 120VAC, 10A, X-PROOF	89395	PAFZZ
E516			1		STARTER, MOTOR	A10EE0AB	(MAIN PUMP MOTOR) NEMA SIZE 3, 3POLE, 460VAC, 120VAC COIL, MAN. RESET, AUX. CONTACT X-PROOF	55841	XB
E517			3		HEATER, OVERLOAD	H1048	(MAIN MOTOR STARTER) 42AMP. NOMINAL	55841	PAFZZ
E518			1		RELAY	KHAU17A11-120	(UUT ISOLATION) 4FORMC, 4PDT, 120VAC, 3A	77342	PAFZZ
E518-1			1		SOCKET	27E166		77342	PAFZZ
E519			1		RELAY	KHAU17A11-120	(UUT ISOLATION) 4FORMC, 4PDT, 120VAC, 3A	77342	PAFZZ
E519-1	V		1		SOCKET	27E166		77342	PAFZZ

PARTS LIST	DESIGN ACTIVITY: APS SYSTEMS 3535 WEST FIFTH STREET OXNARD, CA 93030-6498	CODE IDENT. 60984	PL 20071	CONTRACT NO. F41608-88-C-2452
			SHEET 9 OF 17 SHEETS	SPECIFICATION PD-T-84-005A
LIST TITLE	TEST STAND, RATE OF FLOW		REVISION LETTER/DATE 04/30	AUTHENTICATION / REV / REV / REV

ITEM NO.	DWG. NO.	QTYREQD	-10	SYM	NOMENCLATURE	IDENTIFYING NO.	MATERIAL/SPECIFICATION	FSCM	SMR CODE
E520	20071		1		RELAY, TIMER	CB-1023B78	(RESIDUE TRANSFER) 120VAC,0.6-60 SECS TDDO	77342	PAFZZ
E520-1			1		SOCKET	27E123		77342	PAFZZ
E521			1		RELAY	KHU17D16-24	(FLOW SELECT) 4PDT, 24VDC, 1A GOLD FLASHED CONTACTS	77342	PAFZZ
E521-A			1		SOCKET	27E166		77342	PAFZZ
E521-1			1		DIODE	IN4002	(RELAY COIL SUPPRESSION)	04713	PAFZZ
E522			1		THERMOCOUPLE	NEPB-CPSS-116G-2	(FUEL TEMPERATURE) TYPE 'T', X-PROOF 1/16 DIA X 2 LG	29907	PAFZZ
E523			1		TRANSDUCER, FLOW	QAF-12-WRL-1SC(TC)	(LOW FLOW) TURBINE TYPE, TEMP. COMP., RANGE 50-25,000 PPH, X-PROOF	16951	PBFFF
E524			1		TRANSDUCER, FLOW	QAF-32-WRL-1SC(TC)	(HIGH FLOW) TURBINE TYPE,TEMP. COMP., RANGE 350-120,000 PPH, X-PROOF	16951	PBFFF
E524-1			1		SENSOR, TEMPERATURE	PART OF E524	(TEMPERATURE COMPENSATION)	16951	PBFFF
E525	V		1		TRANSDUCER I/P	T5220-4	(BACK PRESSURE) 4-20mA, 3-15 PSI WITH BOOSTER	07282	XB

PARTS LIST	DESIGN ACTIVITY: APS SYSTEMS 3535 WEST FIFTH STREET OXNARD, CA 93030-6498	CODE IDENT. 60984	PL 20071	CONTRACT NO. F41608-88-C-2452
			SHEET 10 OF 17 SHEETS	SPECIFICATION PD-T-84-005A
LIST TITLE	TEST STAND, RATE OF FLOW		REVISION LETTER/DATE 04/30	AUTHENTICATION / REV / REV / REV

ITEM NO.	DWG. NO.	QTYREQD	QTYREQD -10	SYM	NOMENCLATURE	IDENTIFYING NO.	MATERIAL/SPECIFICATION	FSCM	SMR CODE
E526 THRU E528	20071				NOT USED				
E529			1		TRANSDUCER, PRESSURE	1082AK2YA-10FT.	(BACK PRESSURE) 0-75PSIG, +/- 1/2% ACCY. 2mV/V, -4 PORTS	27385	XB
E530			1		TRANSDUCER, DIFF. PRESSURE	1202AF5YADN	(DIFFERENTIAL PRESSURE) 0-30 PSID, +/- 1/4% ACCY. 2mV/V, -4 PORTS	27385	PBFFF
E531			1		TRANSDUCER, PRESSURE	1082AK2YA-10FT.	(INLET PRESSURE) 0-75PSIG, +/- 1/2% ACCY. 2mV/V, -4 PORTS	27385	XB
E532			1		FUSE, 2 AMP	MDA-2	(D.C. POWER SUPPLY) 300VAC, 2A SLO-BLO 1/4 X 1-1/4	71400	PAFZZ
E533			1		TRANSDUCER I/P	T5220-4	(COOLING WATER) 4-20mA,3-15PSI WITH BOOSTER	07282	PAFZZ
E534			1		TRANSFORMER	2401BE	(UUT POWER) 115/26VAC, 400 HZ	08004	PAFZZ
E535			1		POWER SUPPLY, D.C.	HB24-1.2A	(D.C. EXCITATION) 115VAC/24VDC 1.2A	54407	XB
E536			1		VALVE, SOLENOID	8319C2	(BYPASS VALVE) ALUMINUM BODY, 3/8NPT 120VAC - BANK OF 6 EACH	04845	XB
	V								

PARTS LIST	DESIGN ACTIVITY: APS SYSTEMS 3535 WEST FIFTH STREET OXNARD, CA 93030-6498	CODE IDENT. 60984	PL 20071	CONTRACT NO. F41608-88-C-2452
			SHEET 11 OF 17 SHEETS	SPECIFICATION PD-T-84-005A
LIST TITLE	TEST STAND, RATE OF FLOW		REVISION LETTER/DATE 04/30	AUTHENTICATION	REV	REV	REV

ITEM NO.	DWG. NO.	QTYREQD	-10	SYM	NOMENCLATURE	IDENTIFYING NO.	MATERIAL/SPECIFICATION	FSCM	SMR CODE
E537	20071		1		VALVE, SOLENOID	8319C2	(MASTER FLOW SELECT) ALUMINUM BODY, 3/8NPT 120VAC - BANK OF 6 EACH	04845	XB
E538			1		VALVE, SOLENOID	8319C2	(LOW FLOW) ALUMINUM BODY, 3/8 NPT 120VAC - BANK OF 6 EACH	04845	XB
E539			1		VALVE, SOLENOID	8319C2	(MEDIUM FLOW) ALUMINUM BODY, 3/8NPT 120VAC-BANK OF 6 EACH	04845	XB
E540			1		VALVE, SOLENOID	8319C2	(HIGH FLOW) ALUMINUM BODY, 3/8 NPT 120VAC - BANK OF 6 EACH	04845	XB
E541			1		SWITCH, FLOW	V4-2-U	(VENT FAN) SPDT, 120VAC, 10A X-PROOF	89395	PAFZZ
E542			1		RESISTOR	RCR32-361	(VOLTAGE DROPPING) 360 OHM, 1 WATT, CARBON	50139	PAFZZ
E543			1		RESISTOR	L100J500 WITH TYPE 12 BRACKET	(LOADING RESISTOR) 500 OHM, 100 WATT WIREWOUND	44655	PAFZZ
E544			1		RELAY	KUP11A15-120	(MALFUNCTION)DPDT, 120VAC, 10A	77342	PAFZZ
E544-1			1		SOCKET	27E893		77342	PAFZZ
E545	V		1		CONVERTER, SECOND HARMONIC	CV-800HZ	(UUT SIGNAL CONDITIONER) 2ND HARMONIC TO D.C.	60984	PBFFF

PARTS LIST	DESIGN ACTIVITY: APS SYSTEMS 3535 WEST FIFTH STREET OXNARD, CA 93030-6498	CODE IDENT. 60984	PL 20071 / SHEET 12 OF 17 SHEETS	CONTRACT NO. F41608-88-C-2452 / SPECIFICATION PD-T-84-005A
LIST TITLE	TEST STAND, RATE OF FLOW		REVISION LETTER/DATE: 04/30	AUTHENTICATION / REV / REV / REV

ITEM NO.	DWG. NO.	QTYREQD	-10	SYM	NOMENCLATURE	IDENTIFYING NO.	MATERIAL/SPECIFICATION	FSCM	SMR CODE
E546	20071		1		TIMER/COUNTER,	5334B WITH OPTION 060 AND OPTION 908	(UUT SIGNAL CONDITIONING) GPIB CONTROLLED PERIOD AND INTERVAL MEASURE. 060-REAR TERMINAL 908-RACK MOUNTING	50443	PBFFF
E547			1		POWER SUPPLY, A.C.	101T-2222	(UUT EXCITATION) INPUT-115VAC, 60HZ; OUTPUT-115+/-1VAC, 400 +/-.2HZ, 0.95AMP	16067	PBFFF
E547-1			1		OSCILLATOR, PLUG IN	855T-1-400-2222	(POWER SUPPLY) SINGLE PHASE, 400HZ, PLUGS INTO E547	16067	PAFZZ
E548-1			1		SWITCH, LIMIT	10316H2187	(DOOR INTERLOCK) SINGLE ROTARY LEVER TYPE 120VAC, 10AMP	55841	PAFZZ
E548-2			1		SWITCH, LIMIT	10316H2187	(DOOR INTERLOCK) SINGLE ROTARY LEVER TYPE, 120VAC, 10AMP	55841	PAFZZ
E548-3			1		SWITCH, LIMIT	10316H2187	(DOOR INTERLOCK) SINGLE ROTARY LEVER TYPE, 120VAC, 10AMP	55841	PAFZZ
E548-4			1		SWITCH, LIMIT	10316H2187	(FUSIBLE LINK SENSOR) SINGLE ROTARY LEVER TYPE,120VAC,10AMP	55841	PAFZZ
E548-5			4		ROLLER	E50KL548		55841	PAFZZ
E549	V		1		LIGHT	AC1075	(MAINTENANCE) 120VAC, 100W, INCANDESCENT	57797	PAFZZ

PARTS LIST	DESIGN ACTIVITY: APS SYSTEMS 3535 WEST FIFTH STREET OXNARD, CA 93030-6498	CODE IDENT. 60984	PL 20071	CONTRACT NO. F41608-88-C-2452
			SHEET 13 OF 17 SHEETS	SPECIFICATION PD-T-84-005A
LIST TITLE	TEST STAND, RATE OF FLOW		REVISION LETTER/DATE: 04/30	AUTHENTICATION / REV / REV / REV

ITEM NO.	DWG. NO.	QTY REQD	QTY REQD -10	SYM	NOMENCLATURE	IDENTIFYING NO.	MATERIAL/SPECIFICATION	FSCM	SMR CODE
E550	20071		1		FUSE, 10 AMP	MDA-10	(CONTROL POWER) 300V, 10A,SLO-BLO, 1/4 X 1-1/4	71400	PAFZZ
E551			1		FUSE, 10 AMP	MDA-10	(POWER OUTLETS) 300V, 10A,SLO-BLO, 1/4 X 1-1/4	71400	PAFZZ
E551-1			1		FUSE, HOLDER	S8300-3	3 POLE, 300V, 1/4 X 1-1/4	71400	PAFZZ
E552			1		FUSE, 10 AMP	MDA-10	(LIGHTS) 300V, 10A, SLO-BLO, 1/4 X 1-1/4	71400	PAFZZ
E553			1		LIGHT	AC1075	(SINK) 120VAC, 100W, INCANDESCENT	57797	PAFZZ
E554			1		SWITCH	EDSC-2129	(SINK LIGHT) 120VAC, 10A, SPDT X-PROOF	35705	PAFZZ
E555			1		DIODE, ZENER	IN4743	(VOLTAGE DROPPING)	04713	PAFZZ
E556			3		POWER OUTLET STRIP	UL655SL-BS	120VAC,6-OUTLET WITH CIRCUIT BREAKER/SWITCH	63590	PAFZZ
E557			1		FUSE, 3 AMP	MDA-3	300VAC, 3A, SLO-BLO, 1/4 X 1-1/4	71400	PAFZZ
E558			1		FUSE, 3 AMP	MDA-3	300VAC, 3A, SLO-BLO, 1/4 X 1-1/4	71400	PAFZZ
E559	V		1		TRANSFORMER	26-B-011780-004	SIGNAL # 88-4	05840	PAFZZ

PARTS LIST	DESIGN ACTIVITY: APS SYSTEMS 3535 WEST FIFTH STREET OXNARD, CA 93030-6498	CODE IDENT. 60984	PL 20071	CONTRACT NO. F41608-88-C-2452
			SHEET 14 OF 17 SHEETS	SPECIFICATION PD-T-84-005A

LIST TITLE	REVISION LETTER/DATE				AUTHENTICATION	REV	REV	REV
TEST STAND, RATE OF FLOW	04/30							

ITEM NO.	DWG. NO.	QTYREQD	QTYREQD -10	SYM	NOMENCLATURE	IDENTIFYING NO.	MATERIAL/SPECIFICATION	FSCM	SMR CODE
E560	20071		1		ACUATOR, ROTARY	SM-1140/AD-7710	(LOW FLOW) WITH BUILT-IN AMPLIFIER	05840	PBFFF
E561			1		ACTUATOR, ROTARY	SM-1140/AD-7710	(MEDIUM FLOW) WITH BUILT-IN AMPLIFIER	05840	PBFFF
E562			1		AMPLIFIER	AD-7540	USED WITH ITEM E563	05840	PBFFF
E563			1		ACTUATOR, ROTARY	SM-1580	(HIGH FLOW)	05840	PBFFF
E564 THRU E599					NOT USED				
E600			1		MOTOR, VENT FAN	1951K7	(EXHAUST) 360SCFM @ 1.2"H20 115VAC, 1PH., 60HZ, 1/4H.P., X-PROOF	39428	PAFFF
E601			1		MOTOR, ELECTRIC	UM286TSTGS8001BR-W	(MAIN PUMP) NEMA FRAME 286TS, 30H.P., 460VAC, 3PH, 60HZ, 3530RPM, X-PROOF	38151	AFF
E602			3		FUSE, 100 AMP	FRS-R-100	(MAIN MOTOR) 600VAC, 100AMP	71400	PAFZZ
E602-1			1		FUSE, HOLDER	1BR041	(MOTOR PROTECTION) 600VAC, 100AMP, 3 POLE	71400	PAFZZ
E603 THRU E699	V				NOT USED				

PARTS LIST	DESIGN ACTIVITY: APS SYSTEMS 3535 WEST FIFTH STREET OXNARD, CA 93030-6498	CODE IDENT. 60984	PL 20071 / SHEET 15 OF 17 SHEETS	CONTRACT NO. F41608-88-C-2452 / SPECIFICATION PD-T-84-005A
LIST TITLE	TEST STAND, RATE OF FLOW		REVISION LETTER/DATE 04/30	AUTHENTICATION / REV / REV / REV

ITEM NO.	DWG. NO.	QTYREQD	-10	SYM	NOMENCLATURE	IDENTIFYING NO.	MATERIAL/SPECIFICATION	FSCM	SMR CODE
E700	20071		1		COMPUTER	TRANS286	(SYSTEM CONTROL) 1 MEG FLOPPY DRIVE, 20 MB HARD DISK, MS DOS 3.3 AWARD BIOS	TRANS-COMPU-TER	PBFFF
E700-1					NOT USED				
E700-2					NOT USED				
E700-3			1		CARD, RS232 INTERFACE	FAST COM4	4-CHANNEL	0AYU9	PAFZZ
E700-4			1		CARD, IEEE-488 INTERFACE	GPIB-PC2	IBM PC-AT COMPATIBLE WITH SOFTWARE DRIVER FOR DOS 5-1/4" DISK SIZE	64667	PAFZZ
E700-5			1		CARD, PARALLEL I/O	AC5	24BIT I/O ADAPTER FOR IBM PC-AT	55442	PAFZZ
E700-6			1		CONNECTOR, DISK POWER	1-480426-0	4 POSITION PIN	00779	PAFZZ
E700-7			1		FUSE, 2 AMP	MDA-2	(5V POWER) 300VAC, 2A, SLO-BLO 1/4 X 1-1/4	71400	PAFZZ
E701			1		RACK, ANALOG I/O	AICP-IO PACK	INTELLIGENT I/O ADAPTER	53651	PBFFF
E701-1			4		OUTPUT MODULE	AICP-01420	4-20mA OUTPUT	53651	PAFZZ
E701-2	V		7		INPUT MODULE	AICP-IVX05	0-5VDC INPUT	53651	PAFZZ

PARTS LIST	DESIGN ACTIVITY: APS SYSTEMS 3535 WEST FIFTH STREET OXNARD, CA 93030-6498	CODE IDENT. 60984	PL 20071	CONTRACT NO. F41608-88-C-2452
			SHEET 16 OF 17 SHEETS	SPECIFICATION PD-T-84-005A
LIST TITLE	TEST STAND, RATE OF FLOW		REVISION LETTER/DATE 04/30	AUTHENTICATION / REV / REV / REV

ITEM NO.	DWG. NO.	QTYREQD	-10	SYM	NOMENCLATURE	IDENTIFYING NO.	MATERIAL/SPECIFICATION	FSCM	SMR CODE
E702	20071		1		RACK, DIGITAL I/O	PB16HC	16 CHANNEL I/O WITH SPARE TERMINALS	55442	PAFZZ
E702-1			6		MODULE, OUTPUT	OAC5	115VAC, 3A, N.O.	55442	PAFZZ
E702-2			2		MODULE, OUTPUT	ODC5	5-60VDC, 3A, N.O.	55442	PAFZZ
E702-3			7		MODULE, INPUT	IAC5	115VAC	55442	PAFZZ
E703			1		MODULE, INTERFACE	5727.2	RIBBON TO SCREW TERMINAL ADAPTER	ODU12	PAFZZ
E704 THRU E899					NOT USED				
E900					ASSEMBLY, CABLE	70018		60984	MFF
E901					ASSEMBLY, CABLE	70019		60984	MFF
E902					ASSEMBLY, CABLE	70020		60984	MFF
E903					ASSEMBLY, CABLE	70036		60984	MFF
E904					ASSEMBLY, CABLE	70037		60984	MFF
E905					ASSEMBLY, CABLE	70044		60984	MFF
E906	V				ASSEMBLY, CABLE	70045		60984	MFF

PARTS LIST	DESIGN ACTIVITY: APS SYSTEMS 3535 WEST FIFTH STREET OXNARD, CA 93030-6498	CODE IDENT. 60984	PL 20071	CONTRACT NO. F41608-88-C-2452
			SHEET 17 OF 17 SHEETS	SPECIFICATION PD-T-84-005A
LIST TITLE	TEST STAND, RATE OF FLOW		REVISION LETTER/DATE: 04/30	AUTHENTICATION	REV	REV	REV

ITEM NO.	DWG. NO.	QTYREQD	-10	SYM	NOMENCLATURE	IDENTIFYING NO.	MATERIAL/SPECIFICATION	FSCM	SMR CODE
E907	20071				ASSEMBLY, CABLE	70046		60984	MFF

CHAPTER XIII

COMMERCIAL LITERATURE

13-1 GENERAL

This chapter includes commercial literature and is arranged numerically in sequence, according to item numbers. It is placed in Volumes II and III for Electrical Commercial Literature and Volume IV has the Hydraulic and Mechanical Commercial Literature.

RECOMMENDED MECHANICAL SPARE PARTS LIST

ITEM #	NOMENCLATURE	PART NO.	QTY	MFR
508	Main Filter Element	1300R010P	1	Hycon Corp.
521-1	Air Filter Element	2992-18	1	Norgren
522-2	Coalescing Filter Element	5350-08	1	Norgren
524	Fill Filter Element	0110D010BH/HC	1	Hycon Corp.

RECOMMENDED ELECTRICAL/ELECTRONIC SPARE PARTS LIST

ITEM #	NOMENCLATURE	PART NO.	QTY	MFR
E11, E21	Lamp	BIA(NE-51)	5	Dialight Corp.
E18	Lamp	755	2	Cutler Hammer
E500	Fuse, 15 Amp.	MDA-15	2	Bussmann
E501, E550, E551, E552	Fuse, 10 Amp.	MDA-10	5	Bussmann
E517	Overload Heater	H1048	3	Cutler Hammer
E518, E519	Relay	KHAU17A11-120	2	Potter & Brumfield
E520	Timer Relay	CB-1023B78	1	Potter & Brumfield
E521	Relay	KHU17D16-24	1	Potter & Brumfield
E532, E700-7	Fuse, 2 Amp.	MDA-2	2	Bussmann
E544	Relay	KUP11A15-120	1	Potter & Brumfield
	<u>Second Harmonic Converter</u>			
E545	-Signal Conditioner Board	CV800-A2	1	SATE c/o APS Systems
	<u>A.C. Power Supply</u>			
E547-1	-Plug In Oscillator	855T-1-400-2222	1	California Inst.
E577, E558	Fuse, 3 Amp.	MDA-3	2	Bussmann
E560, E561	<u>Rotary Actuator With Built In Amplifier</u>			
	-Power Supply	68B-025512-001	1	Foxboro/ Jordan
	-Input Card	68A-025499-001	1	Foxboro/ Jordan
	-Amplifier Card	68A-025496-001	1	Foxboro/ Jordan
E562	<u>Amplifier</u>			
	-Top Board	68D-014439-001	1	Foxboro/ Jordan
	-Bottom Board	68C-014248-001	1	Foxboro/ Jordan
	-Output Transistors	31B-014062-001	4	Foxboro/ Jordan

ITEM #	NOMENCLATURE	PART NO.	QTY	MFR
E602	Fuse, 100 Amp.	FRS-R-100	3	Bussmann
	Computer			
E700-3	-RS232 Interface Card	FAST COM4	1	Industrial Computer Source
E700-4	-IEEE-488 Interface Card	GPIB-PC2	1	National Instruments Corporation
E700-5	-Parallel I/O Card	AC5	1	OPTO 22
	Analog I/O Rack			
E701-1	-Output Module	AICP-01420	2	Action Inst.
E701-2	-Input Module	AICP-IVX05	2	Action Inst.
	Digital I/O Rack			
E702-1	-Output Module	OAC5	2	OPTO 22
E702-2	-Output Module	ODC5	1	OPTO 22
E702-3	-Input Module	IAC5	2	OPTO 22

TABLE OF CONTENTS

SOFTWARE DESCRIPTION

FUEL RATE OF FLOW TEST STAND

APS SYSTEMS

Oxnard CA, 93030

GENERAL

Software for the Fuel Rate of Flow Test Stand #10047-10-xx is executed by an IBM-PC/AT computer system running MSDOS Version 3.3 and is written in Turbo C language using the small model. Upon power up the system boots from an EPROM electronic disk, simulating floppy disk drive A:, where the DOS operating system is initially loaded from two system hidden files and the COMMAND.COM file. During boot, an installable device driver is loaded from the boot disk to handle the IEEE-488 communications. An AUTOEXEC.BAT file on the boot disk drive then loads MID, a terminate-and-stay-resident driver from the boot disk that handles RS232 interrupt driven communications. The AUTOEXEC.BAT file then executes the FUEL.EXE program which controls the test stand. This process takes approximately 10 seconds after power up, and allows other intelligent peripheral I/O devices to initialize themselves before starting communications.

If the FUEL.EXE program exits operation for any reason (usually a system configuration error) the system returns to the DOS "prompt" mode (invisible to the operator) where it stays latch up until the computer is reset or until power is recycled. During normal operation with the electronic cabinet closed, the computer cannot be reset. The only way to restart the program then is to turn off the stand power for a few seconds.

The device drivers are:

GPIB.COM — National Instruments PC-II driver configured for "HP5334" at address 03 and "NI5510" at address 15. GPIB.COM is loaded from the CONFIG.SYS file.

MID.COM — Parasoft RS232 interrupt driver for serial ports COM1 through COM4 implemented using the FASTCOM-4 hardware interface card. This driver is preconfigured with all ports set to 9600 Baud, 8 Bits, No Parity, 1 Stop, no handshaking. Preconfiguration information is stored in the file MID.CF, which must be located in the root or active directory when MID.COM is executed. When executed later, the FUEL.EXE control program then reconfigures each port upon initialization to the required parameters for the particular RS232 device. MID is loaded from AUTOEXEC.BAT.

FUEL.EXE PROGRAM

The software control program FUEL.EXE requires approximately 80 Kbytes of DOS memory to execute. The program loads UUT, Calibration, and system configuration data from the read/write files UUT.DAT, CAL.DAT, and SYS.DAT respectively. These data files reside on a non-volatile electronic RAM disk simulating hard disk drive C:. As new UUT's are added to the test stand configuration, the FUEL.EXE program updates the UUT.DAT file using the standard DOS file manipulation calls. Likewise recalibration of the flow control, or changing of system constants, will cause their respective data files to be updated.

COMPILATION

The program is compiled using Turbo-C with small model libraries. A custom header FUEL.H includes definitions of functions, equates, structures, port assignments, and global variables. The main module initializes all variables and then loops continuously, calling various functional modules depending on the mode of operation. Most functions that require time to complete their processing are broken into steps, with only a single step performed at a time. Between steps, program control is returned to the MAIN module to allow updating of other modules. Thus, simplified multi-tasking is implemented by time sharing the processor.

MODES OF OPERATION

The five basic modes of operation are:

- STANDBY: Setup of UUT parameters & configuration of equipment.
- AUTOMATIC: Normal mode of operation. Automatically controls flow according to operator input. Automatically reads and displays UUT and master flow. Automatically controls UUT Response tests.
- MANUAL: Similar to AUTOMATIC mode but also automatically controls backpressure according to operator input.
- REMOTE: Allows control of flow with an external input 0 to 5 VDC signal. Backpressure and all other control functions are disabled. UUT flow and master flow are continuously updated.
- PROGRAM: Non testing mode for programming UUT and SYSTEM data, and calibrating flow and backpressure control data.

MAIN MODULE

After initialization of variables and control functions, the main software test loop is continuously executed until an unrecoverable system error is encountered. System faults are broken into the following two categories:

```
system_error    integer, DOS data file read & write errors,
                    terminates program
sys_message     ascii string message indicating type of fault,
                    does not terminate program
```

The DOS tick clock is used to schedule iterative operation, through the use of variable time which is set to the present clock count at the beginning of each main loop. The rest of the modules in the main loop are, or are not, executed depending on the TEST MODE at the time. The following documents the basic flow during the various test modes.

BASIC TEST MODE MODULES

STANDBY MODE: The first time control changes from another mode to the standby mode, flag system_mode is not equal to flag last_sys, and the following initializations are performed:

UUT maximum flow is checked and the appropriate master flowmeter is set using master_small() or master_large().
Terminal #2 is setup for displaying sys_message
Terminal #1 is setup to display flow
Terminal #2 is pre-setup to enter flow values when mode changes
Flow is set to zero
Variable newval, the latest value entered on terminal #2, is set to zero
Flag message_mode, used by terminal #2 to display sys_message, is set true
Flow is bypassed around the UUT
Flag horn is set off to disable the audible horn in alarm_proc()

Prog_mode() is then processed as described in PROGRAM MODE below. If the keyswitch is set for programming then uut_select() is not executed. Otherwise uut_select() monitors terminal #2 and builds an operator entered string which is converted to a test code number. Uut_select() always calls module setup_display() which displays on terminal #1 any of the setup information messages stored for the presently selected UUT code. The get_setup_msg() module is used to copy the appropriate setup message for display on terminal #1.

AUTOMATIC MODE: During every automatic mode main loop displ_service() updates terminal #1 information according to the latest commanded key from terminal #1. Each time the module is entered it attempts to input a character from terminal #1. If a character is entered, it immediately sets integer variable displ_mode to the proper value to display the requested information. If a character is received, update_displ flag is set and terminal #1 is updated with the appropriate information.

Module master_flow() is then executed which reads any new characters sent from the master flow meter and appends these characters to previously received characters. If a terminator (line feed) is received the message is decoded into a flow value, the value added to a first-in-first-out stack of 6 values, the last six values are averaged, variable sys_flow is updated with the average for use by the rest of the program, and the module returns with a flag set indicating an update. The number of entries to master_flow() between updates is counted and if greater than 20000 (approx 10 sec) an alarm flag is set. Also if more than 42 characters are received without a terminator, then the character string is discarded.

Whenever the master flow meter value is updated, uut_flowcalc() is skipped to limit the amount of processing time in a single main loop. Otherwise uut_flowcalc() is executed to continuously update the present UUT angle information. The processing used depends on the UUT TYPE code entered in the UUT DATA for the various transducers. For three phase synchro type UUT's, a single GPIB measurement is made. For 800 Hz second harmonic type transducers, a single measurement of the converters analog output voltage (multiplied by the amplitude of 115VAC 400 Hz power) is used to calculate the angle. For the "drum & impeller" type transducer, a sequence of GPIB commands are sent to the timer counter to setup and measure the period and interval of the two pulses, which is then used to calculate angle. To limit processing time, only one step of this process is executed each entry into uut_flowcalc(), based on the value of integer uut_flowmod.

In the AUTOMATIC test mode the variable set_mode is forced to allow only flow (not backpressure) to set and controlled by the operator.

If a "Response Test" is selected by terminal #1, variable control_mode is set to RESPONSE, and response_service() is executed. Response_service() uses variable resp_mod to determine which step of the test should be processed before immediately returning to the main loop. To limit processing time, the remainder of the main loop software is bypassed, and loop is restarted.

If "Freeze" is selected by terminal #1 (during setup for a "response test"), variable control_mode is set to FREEZE, and the master flow meter and terminal #2 are updated per previously defined modules.

If neither Response or Freeze are active, bp_service() and disp2_service() are executed. Bp_service() returns without processing if the test mode is AUTOMATIC. Disp2_service monitors terminal #2 and updates flow control based on operator inputs, in a similar fashion as displ_service described above.

If the system test mode changes, terminals #1 and #2 are reset to initial conditions using reset_displ() and reset_disp2().

MANUAL MODE: The MANUAL test mode process is identical to the automatic test mode except terminal #2 mode is not forced to setting of flows only, but can also set and control backpressure. Bp_service() is called each loop to implement pseudo PID processing. Backpressure control is described below.

EXTERNAL MODE: During EXTERNAL testing mode, displ_service(), master_flow() and uut_flowcalc() are processed, but bp_service(), disp2_service(), response and freeze functions are skipped. External_control() reads the analog voltage at the external input jack and calculates a flow value based on the present UUT maximum flow data value. It also flushes characters from both terminal#1 and #2, which are inactive but can send characters if the operator presses a key.

PROGRAM M0DE: The program mode can be entered only when the test mode switch is in STANDBY. The module program_mode monitors the terminal #2 keypad for commands and uses terminal #1 and #2 for displaying information. Integer flag prog_mode is set true by swith_input() if the keyswitch is in the program position. The integer flag prog_mod is preset to zero to indicate first entry into the programming mode for initialization of variables. After initialization, each entry into the module attempts to read a new character from keypad #2 (DISP2PORT). If a character is received it is immediately processed. Function keys and "ENTER" are immediately acted upon. Other keys are used to build a ascii string for use by one of the function keys.

Module new_prog_data is called whenever a string is completed, to store the new data entered into the proper memory location. When STORE DATA is selected, DOS system calls are used to create a backup of all three data files before writing the new data to the appropriate file name.

FLOW CONTROL

Various parts of the program command a specific flow through the module adjust_flow(). This is the only module which performs its entire processing without returning to the calling program. Embedded delays require up to 3.5 seconds to execute if valve changes are required.

Adjust_flow() outputs the required control currents and valving required to achieve the desired fuel flow. Three overlapping regulating valves with overlapping ranges are required to cover the entire control range of 50 to 125000 PPH. Due to backlash (hysteresis) in the valve controls, a different current is required to achieve a particular flow depending if the flow is being reduced or increased. Therefore two separate calibration tables of flow versus control current are utilized. During automatic flow calibration (PROGRAM mode), one table is determined with increasing flow, and the other determined with decreasing flow.

When a new flow values requires a different valve than previously selected, adjust_flow() sequences through several steps to insure that the transition is not destructive to the UUT. First all three regulating valves are set to their minimum flow and their respective enable valves closed. Fuel flow is diverted around the UUT using the "bypass" function. The present value of backpressure current is temporarily stored, and backpressure current is set to 0 to immediately remove any restriction to the flow. After a 1.5 second delay to let the valves settle, the new regulating valve is enabled by opening its associated pneumatic ball valve. During the enabling of this valve the flow may temporarily (for 1 second) rise to more than 50% of its full rated flow, even with minimum flow selected. During this surge the flow is diverted around the UUT. After an additional 1.5 sec the surge has passed, and the temporarily stored backpressure value is restored, the bypass function is disabled allowing flow through the UUT, and the regulating value is set to its proper value for the desired new flow.

BACKPRESSURE CONTROL

Bp_service() implements a PID process by sampling the backpressure approximately twice a second, calculating the error between desired and actual backpressure, and calculating a new value based on error, the integral of error over time, and the rate of error change from sample to sample. A gain is applied to each PID term which is settable from terminal #2 during programming of SYSTEM DATA. The time between samples is also settable.

Each new value of backpressure calculated by the PID process is passed to bp_set() which calculates the approximate (within 50%) control current required to induce the proper restriction of flow by partially closing valve 520. The current is interpolated from a 17 point table of restriction values determined during the automatic flow calibration in the PROGRAM mode. The table restriction values are very imprecise because of the nonlinearities of the backpressure valves. The induced restriction should basically be a function of backpressure divided by flow-squared. The actual calculation used was arrived at through curve fitting of a large amount of empirical data gathered during the test stand development. The exponent of flow in this calculation is settable as part of SYSTEM DATA, and the value as delivered is documented in the operators manual.

The first time that backpressure is set from terminal #2, bp_service() initializes the integral part of the PID processing with the new value. Note also that whenever the operator requested flow requires regulator valve change, the backpressure restriction must be temporarily removed so that flow surges do not overpressurize the UUT holding fixture. This subject is discussed above in adjust_flow().

CALIBRATION

Calibration is performed as part of the PROGRAM mode. Once called, the cal_flow() module takes complete control of the calibration procedure. During required delays, cal_delay() is called which continuously updates system flow using master_flow(), and monitors switch inputs and alarms for faulty conditions. If the pump is turned off or is disabled due to alarm conditions, cal_delay() will stop all flow and return with a fault flag set. The calling cal_flow() routine immediately tests the flag and returns to PROGRAM mode if a fault was detected.

Cal_flow() first calibrates flow from maximum to minimum. This immediately purges any air in the system that would make low flow measurements inaccurate. The calibration is performed by outputting 17 steps of control current from 16 ma to 0 ma to the presently selected valve. These currents are rescaled for 20 to 40 ma respectively by the ma_out() routine. After waiting 12 seconds for the flow to stablize at each output current, the master flowmeter value is stored in a calibration table. When the table is complete, the process is repeated for flow from minimum to maximum. This data is stored in a separate table. During normal operation the module adjust_flow() uses one or the other of these tables for controlling flow based on increasing or decreasing flow commands.

When minimum to maximum flow calibration is completed, cal_flow() generates a backpressure restriction table by outputting 17 steps of current to the backpressure valves. An iterative process delays to let the valve settle, then measures the backpressure and adjusts slightly the flow to achieve approximately 7 PSI backpressure. The number of iterations is set to four. At the end of the fourth iteration, a restriction value is calculated by dividing the induced backpressure by flow raised to the power of SYSTEM variable bp_exponent. Induced backpressure is measured minus residual (never goes to zero). The restriction value is stored as an entry in the 17 point backpressure table which is part of SYSTEM data.

SYSTEM DATA

The PROGRAMMING mode allows alteration of SYSTEM data stored in the file SYS.DAT. The operator must enter the password "89-123" (plus "ENTER") to gain access to these variables. The following is a list of the variables and their function. TABLE II of the operators manual contains the present values of these variables. It is imperative that this list be updated with any changes to the data.

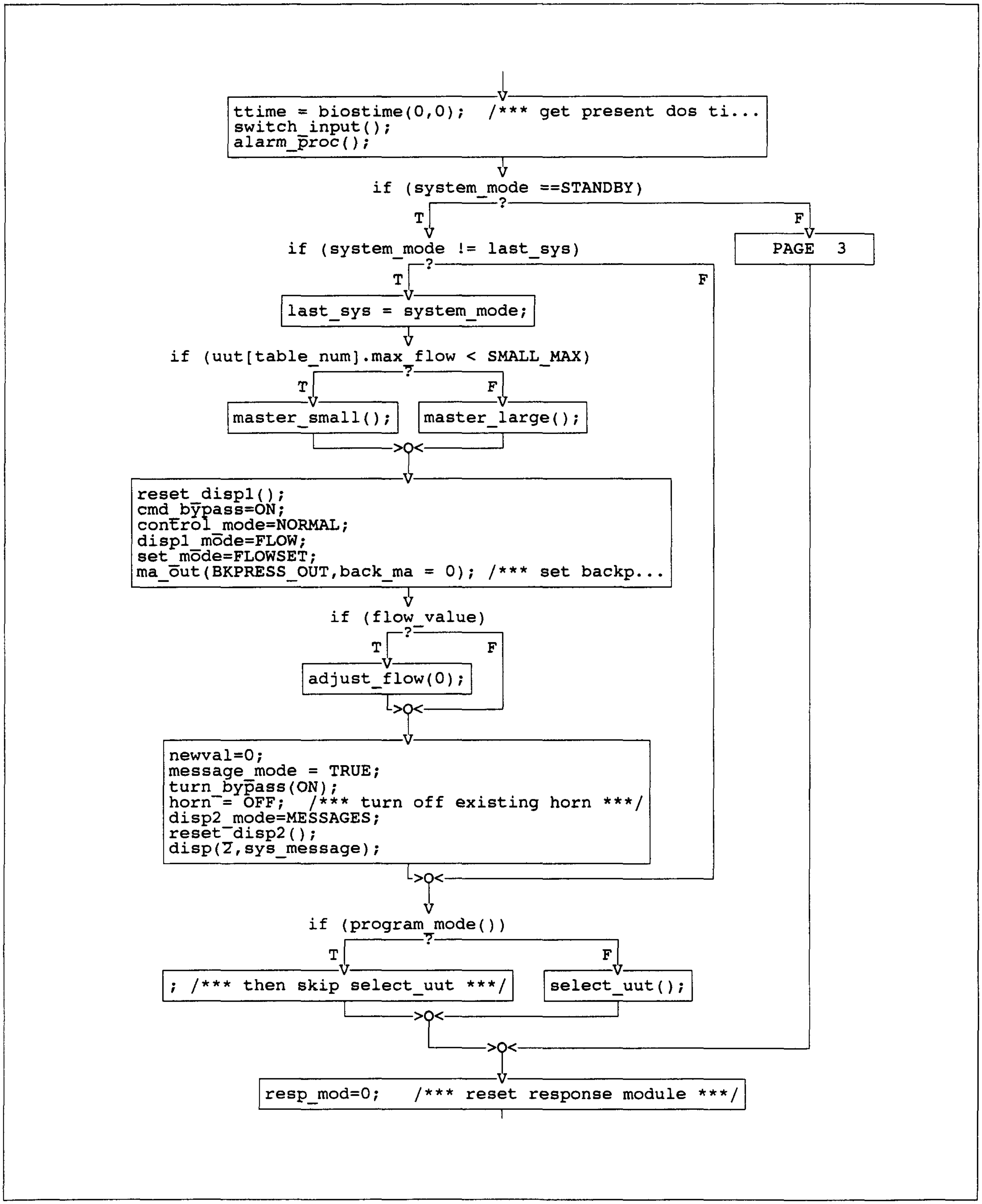
SYSTEM VARIABLES

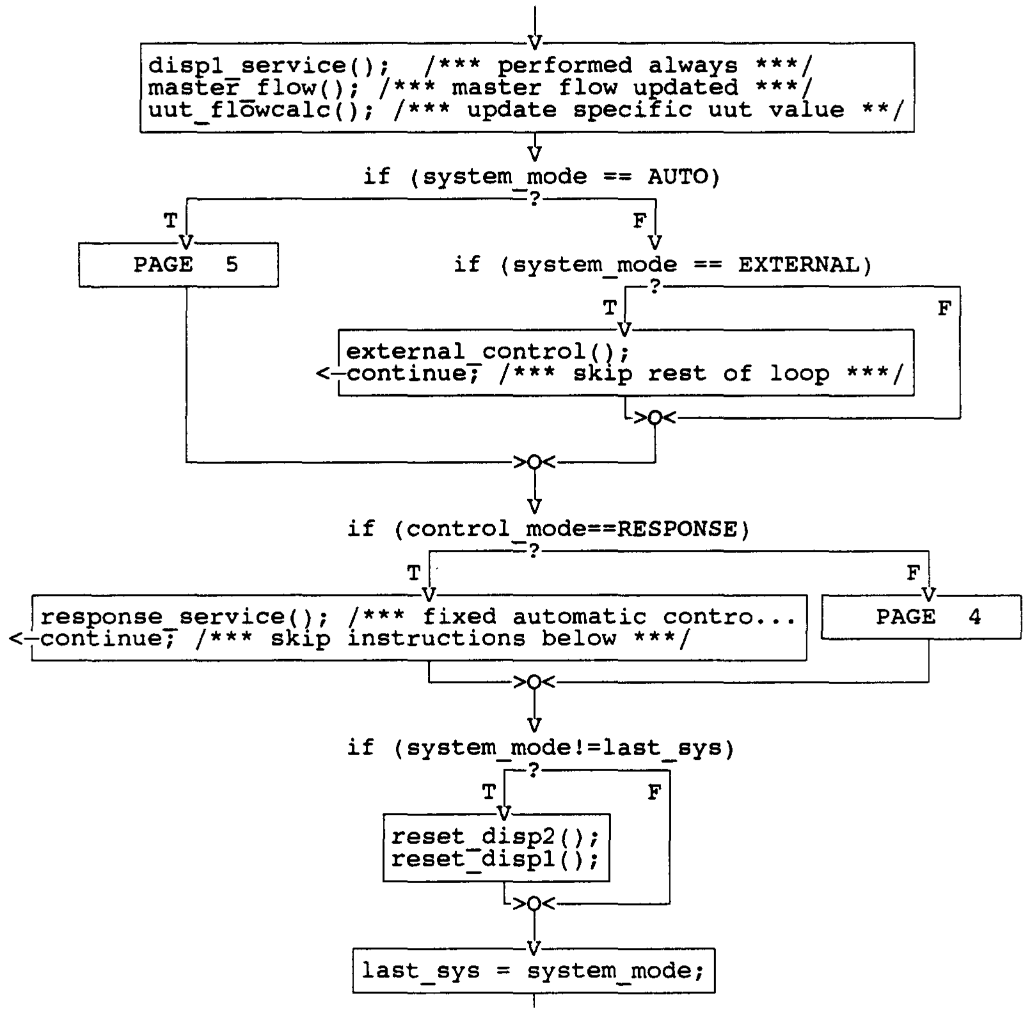
ITEM	DESCRIPTION
SYS bp PID gain	Proportional Gain of PID processing
SYS bp PID integ	Integral Gain of PID processing
SYS bp PID diff	Differential Gain of PID processing
SYS bp PID time	Time between PID processes in 1/18 sec
SYS bp disp gain	Gain applied to E20 convert volts to PSI
SYS bp Exponent	Exponent power of flow for restriction calculations
SYS bp Min Press	Residual backpressure without additional backpressure valve restriction
SYS valve1 max	Not used
SYS valve1 min	Not used
SYS valve2 max	Not used
SYS valve2 min	Not used
SYS valve3 max	Not used
SYS valve3 min	Not used
SYS Harmonc gain	Angle conversion gain for TYPE 1 harmonic UUTs (degree/(volts/115))
SYS Harmnc offst	Angle conversion offset for TYPE 1 harmonic UUTs (degrees)
SYS Drum gain	Angle conversion gain for TYPE 2 UUTs (degrees/degree)
SYS Drum offset	Angle conversion offset for TYPE 2 UUTs (degrees)
SYS Extrnal gain	Multiplier for external volts input during REMOTE mode to achieve max uut flow. Input is 0 to 5 volts, so multiplier should be 1/5.

```
system_error = FALSE;
initialize();      /*** initialize interfaces ***/
cmd_bypass=ON;
bp_enable=OFF;   /* defeat loop */
bp_val=0;
adjust_flow(1);     /* set essentially no flow */
displ_mode=SETUP;
disp2_mode=MESSAGES;
sel_mod1=0;     /* reset uut process # */
sel_mod2=0;     /* reset select test module */
group=UUT;
...
```

```
while (!system_error || prog_mode)
?
T          F
PAGE  2
```

```
adjust_flow(0);
strcpy(msg1,">>>>SYSTEM FAIL<<<<");
disp(1,msg1);
sprintf(msg2,"ERROR CODE %d",system_error);
disp(2,msg2);
#ifdef DEBUG
printf("SYSTEM FAILURE CODE: %d\n",system_error);
#endif
exit (7);
```

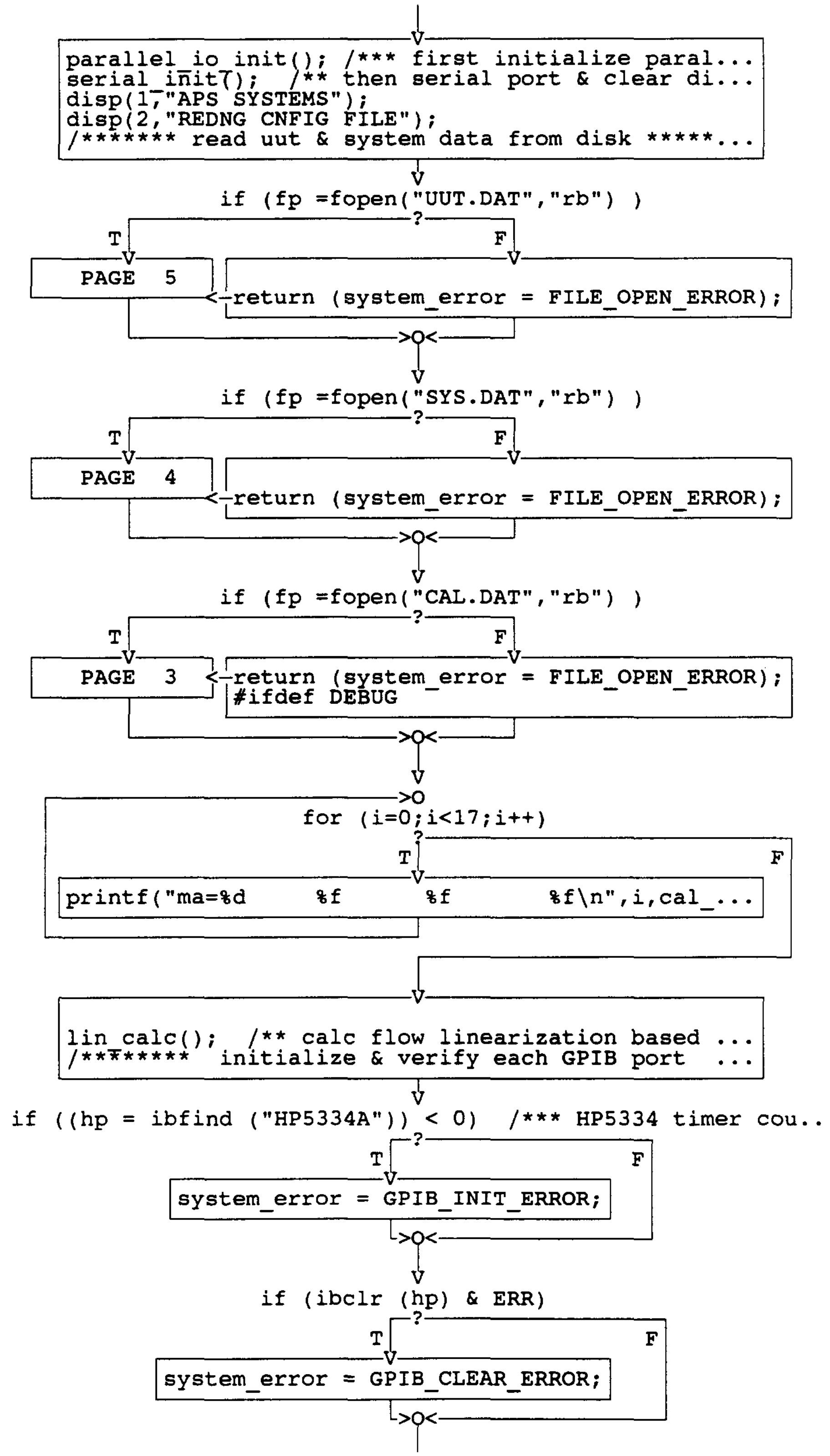

ttime = biostime(0,0); /*** get present dos ti...
switch_input();
alarm_proc();
if (system_mode ==STANDBY)
T
F
PAGE 3
if (system_mode != last_sys)
T
F
last_sys = system_mode;
if (uut[table_num].max_flow < SMALL_MAX)
T
F
master_small();
master_large();
reset_displ();
cmd_bypass=ON;
control_mode=NORMAL;
displ_mode=FLOW;
set_mode=FLOWSET;
ma_out(BKPRESS_OUT,back_ma = 0); /*** set backp...
if (flow_value)
T
F
adjust_flow(0);
newval=0;
message_mode = TRUE;
turn_bypass(ON);
horn = OFF; /*** turn off existing horn ***/
disp2_mode=MESSAGES;
reset_disp2();
disp(2,sys_message);
if (program_mode())
T
F
; /*** then skip select_uut ***/
select_uut();
resp_mod=0; /*** reset response module ***/

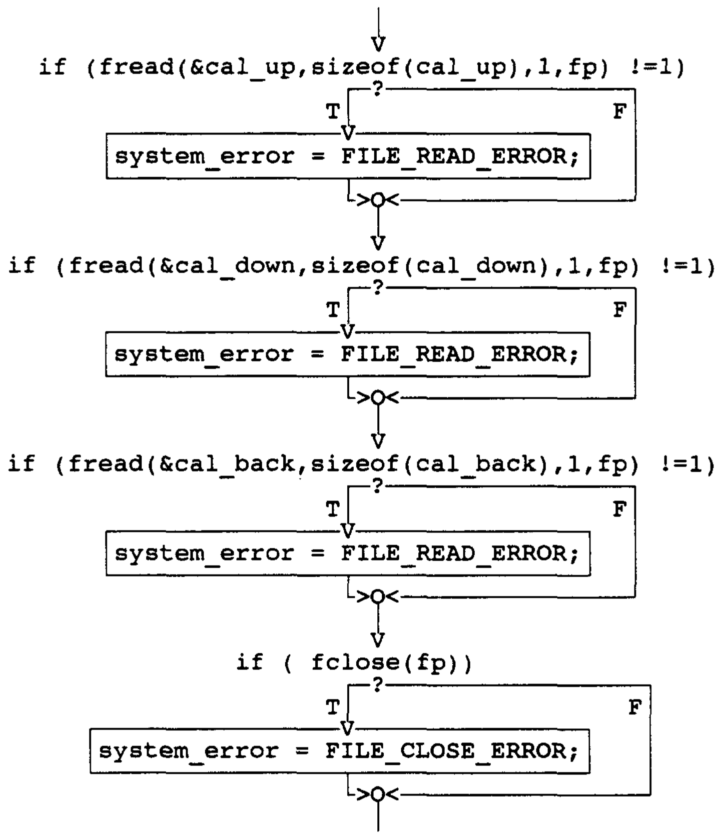

displ_service(); /*** performed always ***/
master_flow(); /*** master flow updated ***/
uut_flowcalc(); /*** update specific uut value **/
if (system_mode == AUTO)
T
F
PAGE 5
if (system_mode == EXTERNAL)
T
F
external_control();
continue; /*** skip rest of loop ***/
if (control_mode==RESPONSE)
T
F
response_service(); /*** fixed automatic contro...
continue; /*** skip instructions below ***/
PAGE 4
if (system_mode!=last_sys)
T
F
reset_disp2();
reset_disp1();
last_sys = system_mode;

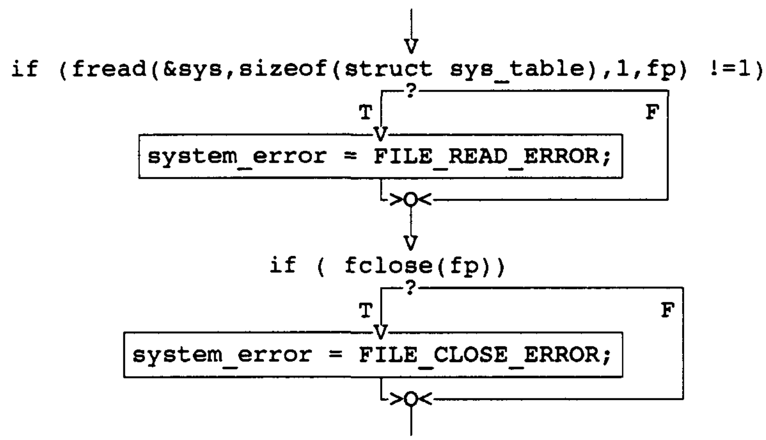
```
if (control_mode==FREEZE)
T                                                        F
master_flow();    /*** flush master flow buffer ...      bp_service(PID_LOOP);
serial_in(DISP2PORT); /**** flush display2 buff...       disp2_service();
```

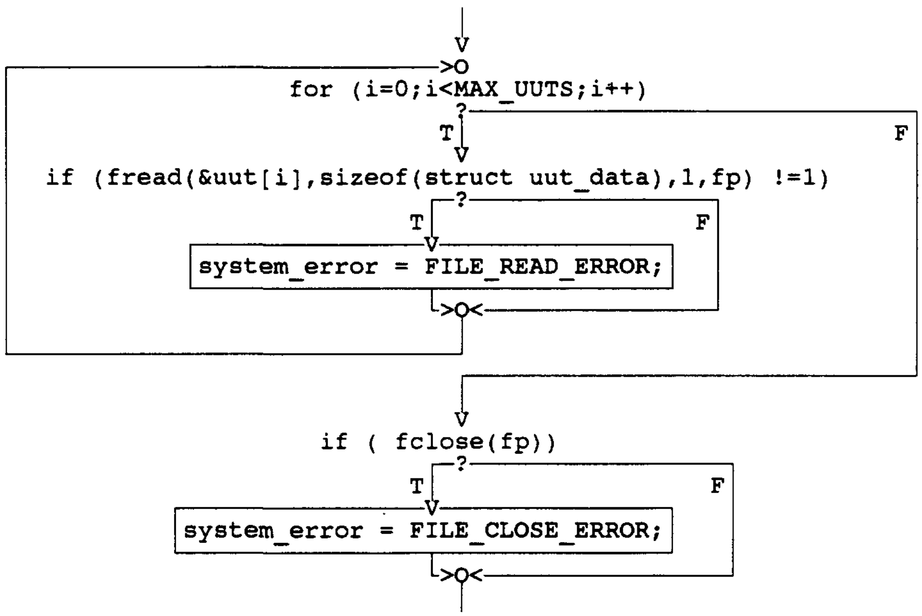
```
set_mode=FLOWSET; /*** limit to flow only ***/
disp2_mode=FLOWSET;
```

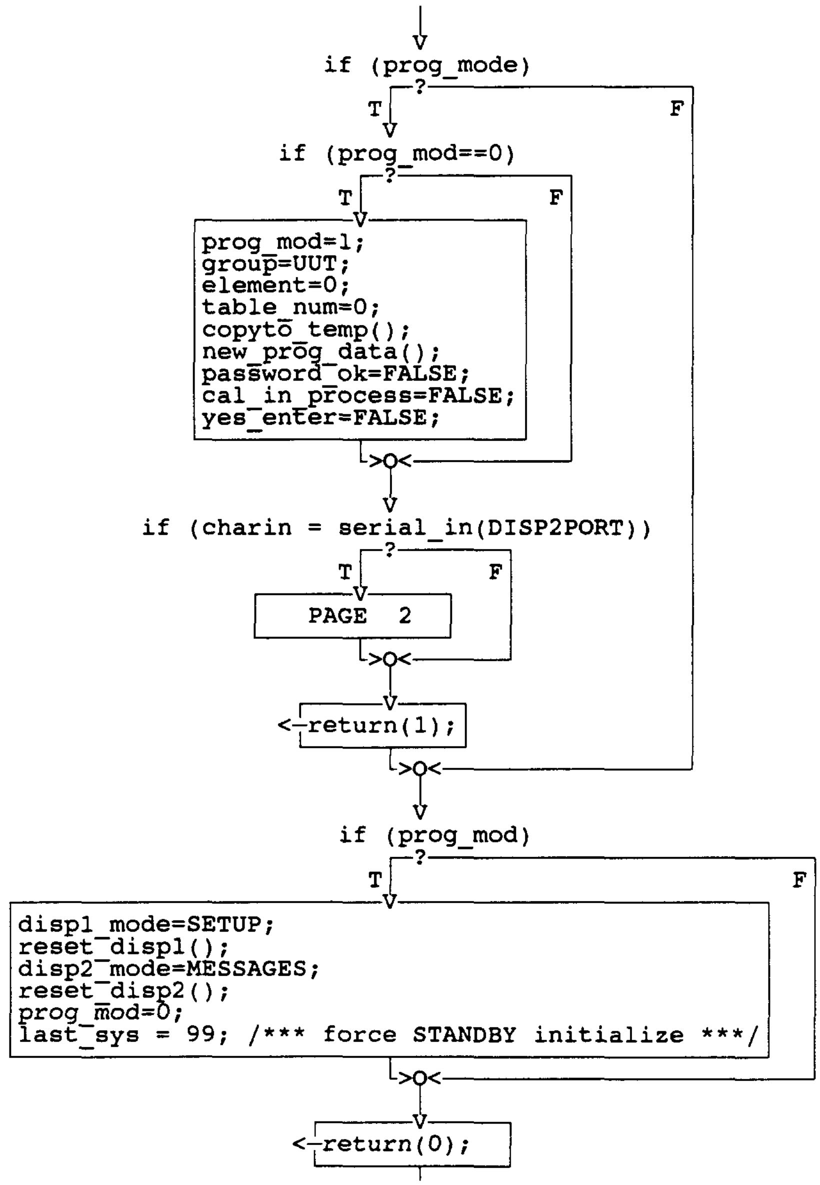
```
PAGE  2

ibwrt (hp,"IN AU0 CO0 SE1 AS1 AX1 BS1 BX1 TR0 F...

if (ibsta & ERR)
  T:
    gpib_fault = TRUE;
    strcpy(sys_message,"GPIB SEND Error");
  F

if ((na8810 = ibfind ("NA8810")) <0)  /*** North Atlantic...
  T:
    system_error = GPIB_INIT_ERROR;
  F

if (ibclr (na8810) & ERR)
  T:
    system_error = GPIB_CLEAR_ERROR;
  F

analog_init(); /*** initialize action pak analo...
/*** calculate backpressure period based on pro...
pid_period = (int)(18 * sys.bp_Time);
```

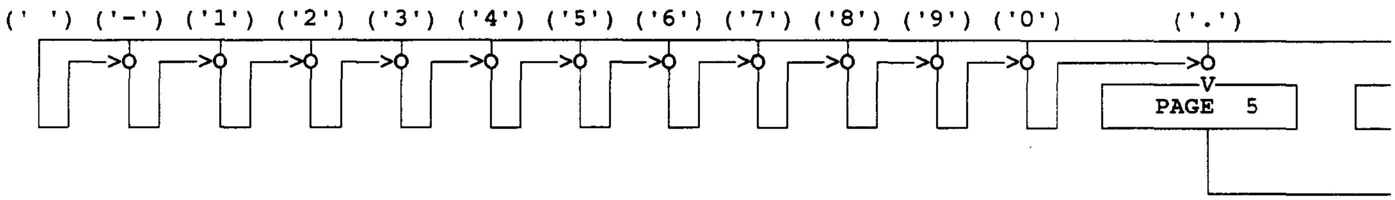

parallel_io_init(); /*** first initialize paral...
serial_init(); /** then serial port & clear di...
disp(1,"APS SYSTEMS");
disp(2,"REDNG CNFIG FILE");
/******* read uut & system data from disk *****...
if (fp =fopen("UUT.DAT","rb"))
T
F
PAGE 5
return (system_error = FILE_OPEN_ERROR);
if (fp =fopen("SYS.DAT","rb"))
T
F
PAGE 4
return (system_error = FILE_OPEN_ERROR);
if (fp =fopen("CAL.DAT","rb"))
T
F
PAGE 3
return (system_error = FILE_OPEN_ERROR);
#ifdef DEBUG
for (i=0;i<17;i++)
T
F
printf("ma=%d %f %f %f\n",i,cal_...
lin_calc(); /** calc flow linearization based ...
/******** initialize & verify each GPIB port ...
if ((hp = ibfind ("HP5334A")) < 0) /*** HP5334 timer cou...
T
F
system_error = GPIB_INIT_ERROR;
if (ibclr (hp) & ERR)
T
F
system_error = GPIB_CLEAR_ERROR;

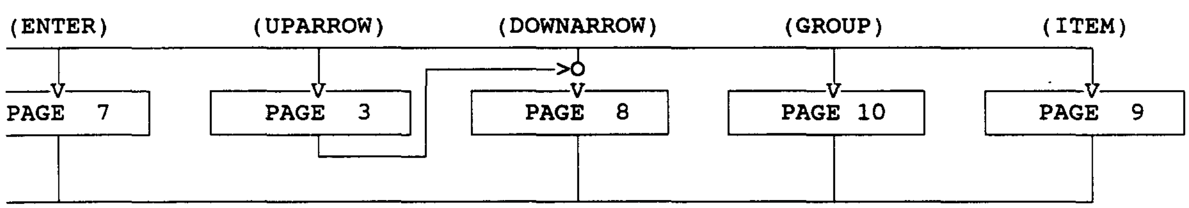

if (fread(&cal_up,sizeof(cal_up),1,fp) !=1)
?
T
F
system_error = FILE_READ_ERROR;
if (fread(&cal_down,sizeof(cal_down),1,fp) !=1)
?
T
F
system_error = FILE_READ_ERROR;
if (fread(&cal_back,sizeof(cal_back),1,fp) !=1)
?
T
F
system_error = FILE_READ_ERROR;
if (fclose(fp))
?
T
F
system_error = FILE_CLOSE_ERROR;


if (fread(&sys,sizeof(struct sys_table),1,fp) !=1)
?
T
F
system_error = FILE_READ_ERROR;
if (fclose(fp))
?
T
F
system_error = FILE_CLOSE_ERROR;


for (i=0;i<MAX_UUTS;i++)
T
F
if (fread(&uut[i],sizeof(struct uut_data),1,fp) !=1)
T
F
system_error = FILE_READ_ERROR;
if (fclose(fp))
T
F
system_error = FILE_CLOSE_ERROR;

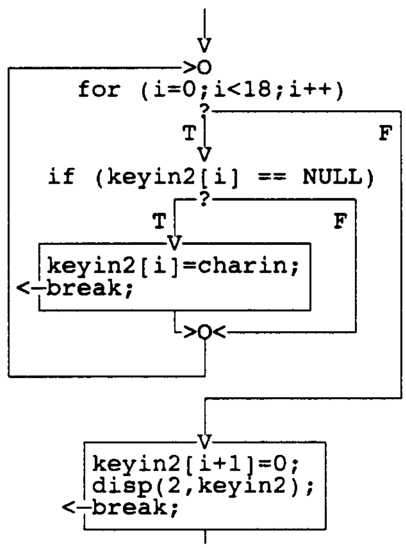

if (prog_mode)
T
F
if (prog_mod==0)
T
F
prog_mod=1;
group=UUT;
element=0;
table_num=0;
copyto_temp();
new_prog_data();
password_ok=FALSE;
cal_in_process=FALSE;
yes_enter=FALSE;
if (charin = serial_in(DISP2PORT))
T
F
PAGE 2
return(1);
if (prog_mod)
T
F
disp1_mode=SETUP;
reset_disp1();
disp2_mode=MESSAGES;
reset_disp2();
prog_mod=0;
last_sys = 99; /*** force STANDBY initialize ***/
return(0);

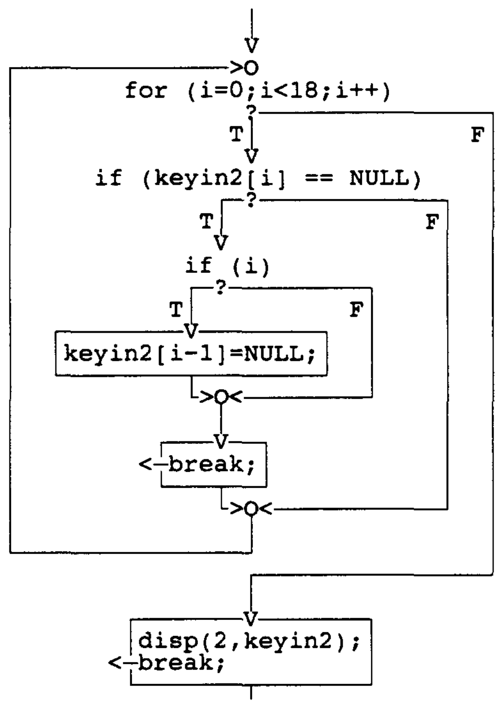

(' ') ('-') ('1') ('2') ('3') ('4') ('5') ('6') ('7') ('8') ('9') ('0') ('.')
PAGE 5

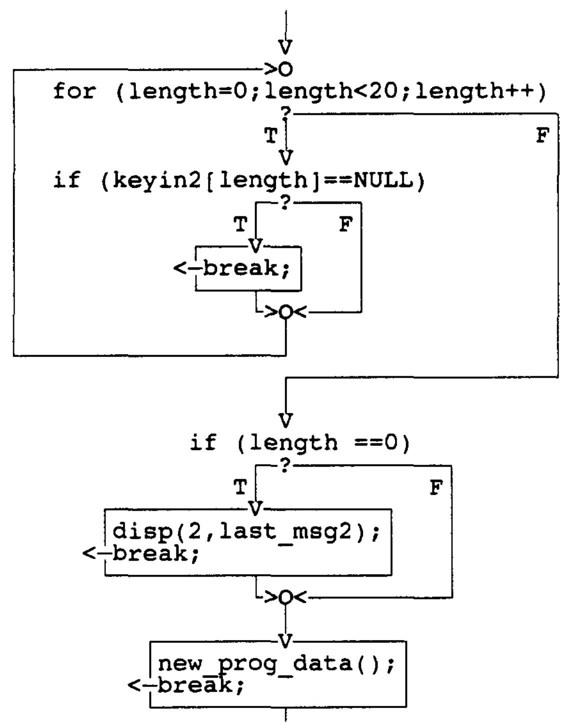

(ENTER)
(UPARROW)
(DOWNARROW)
(GROUP)
(ITEM)
PAGE 7
PAGE 3
PAGE 8
PAGE 10
PAGE 9

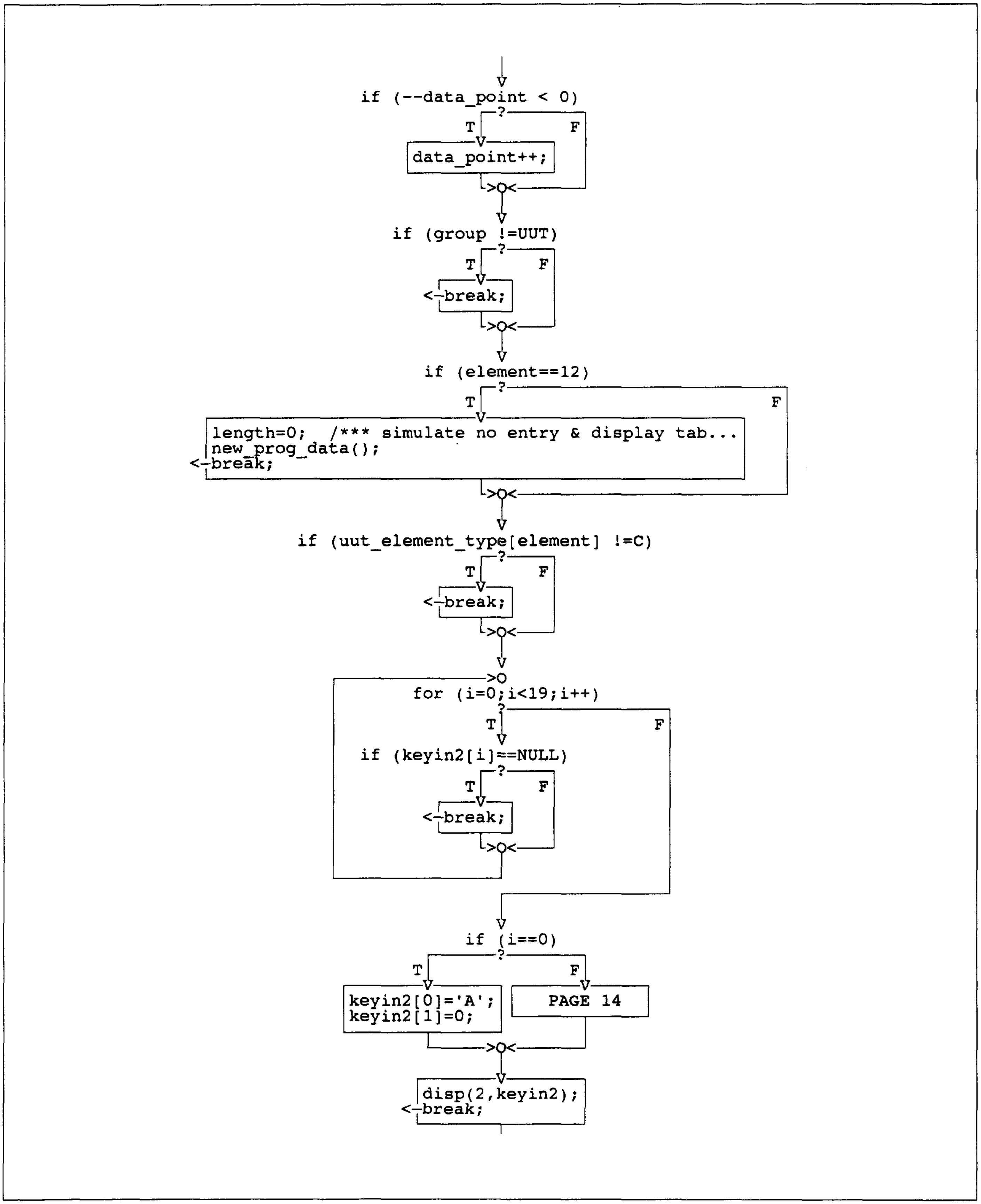
```
data_point+=2;
```

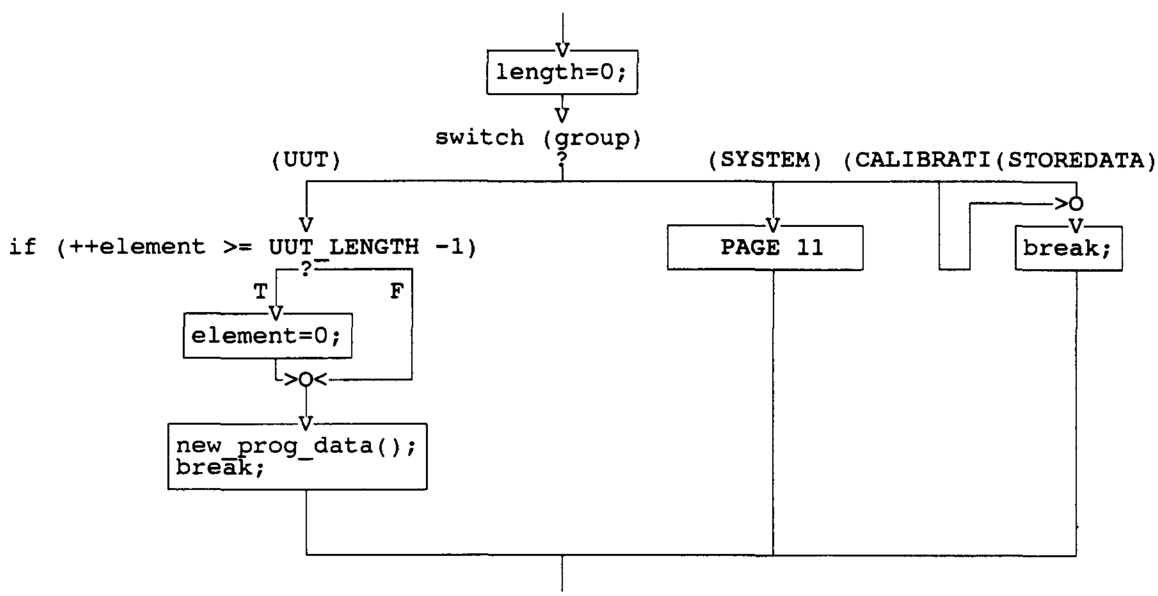
```
keyin2[0]=NULL;
disp(2,"");
<-break;
```

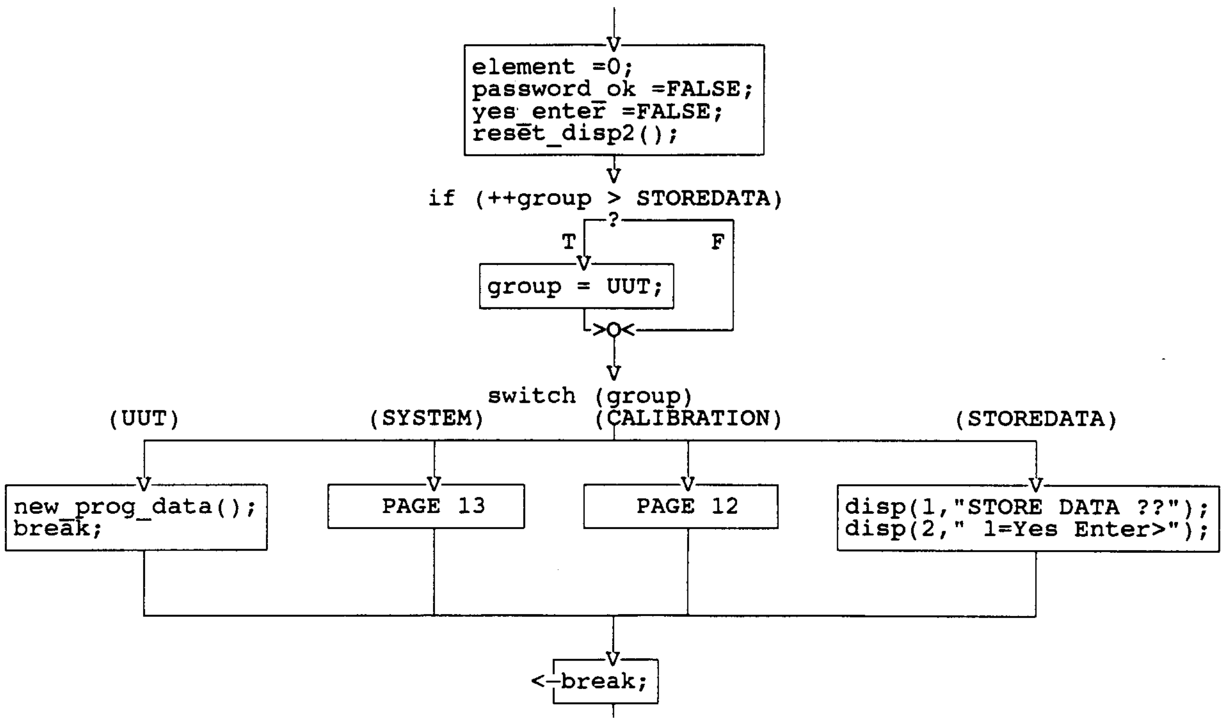

for (i=0;i<18;i++)
T
F
if (keyin2[i] == NULL)
T
F
keyin2[i]=charin;
break;
keyin2[i+1]=0;
disp(2,keyin2);
break;

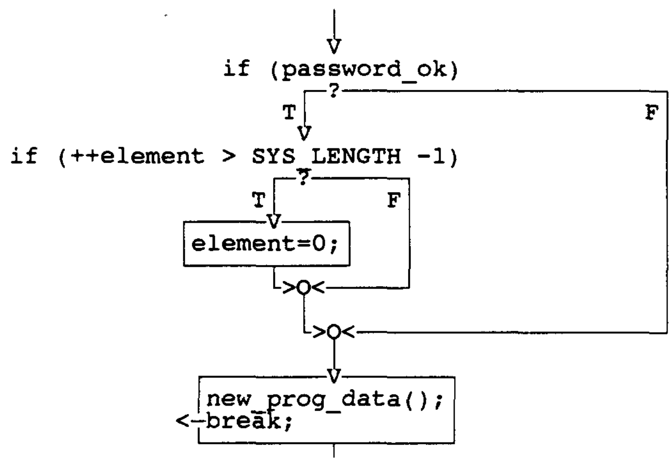

for (i=0;i<18;i++)
T
F
if (keyin2[i] == NULL)
T
F
if (i)
T
F
keyin2[i-1]=NULL;
break;
disp(2,keyin2);
break;


for (length=0;length<20;length++)
?
T
F
if (keyin2[length]==NULL)
?
T
F
break;
if (length ==0)
?
T
F
disp(2,last_msg2);
break;
new_prog_data();
break;


if (--data_point < 0)
T
F
data_point++;
if (group !=UUT)
T
F
break;
if (element==12)
T
F
length=0; /*** simulate no entry & display tab...
new_prog_data();
break;
if (uut_element_type[element] !=C)
T
F
break;
for (i=0;i<19;i++)
T
F
if (keyin2[i]==NULL)
T
F
break;
if (i==0)
T
F
keyin2[0]='A';
keyin2[1]=0;
PAGE 14
disp(2,keyin2);
break;

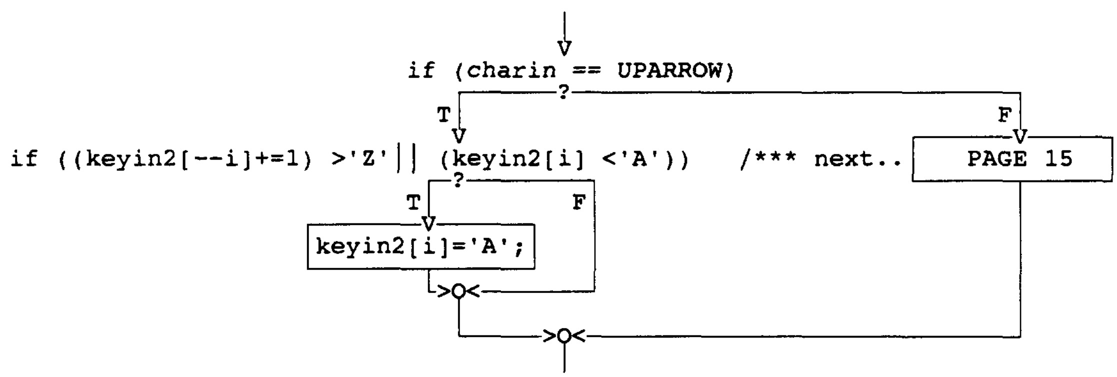

length=0;
switch (group)
(UUT)
(SYSTEM)
(CALIBRATI
(STOREDATA)
if (++element >= UUT_LENGTH -1)
T
F
element=0;
new_prog_data();
break;
PAGE 11
break;

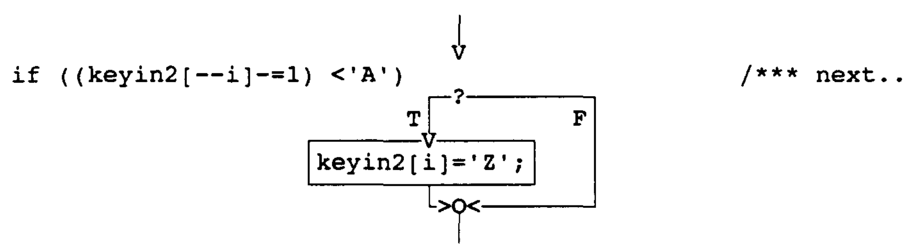

element =0;
password_ok =FALSE;
yes_enter =FALSE;
reset_disp2();
if (++group > STOREDATA)
?
T
F
group = UUT;
switch (group)
(UUT)
(SYSTEM)
(CALIBRATION)
(STOREDATA)
new_prog_data();
break;
PAGE 13
PAGE 12
disp(1,"STORE DATA ??");
disp(2," 1=Yes Enter>");
break;

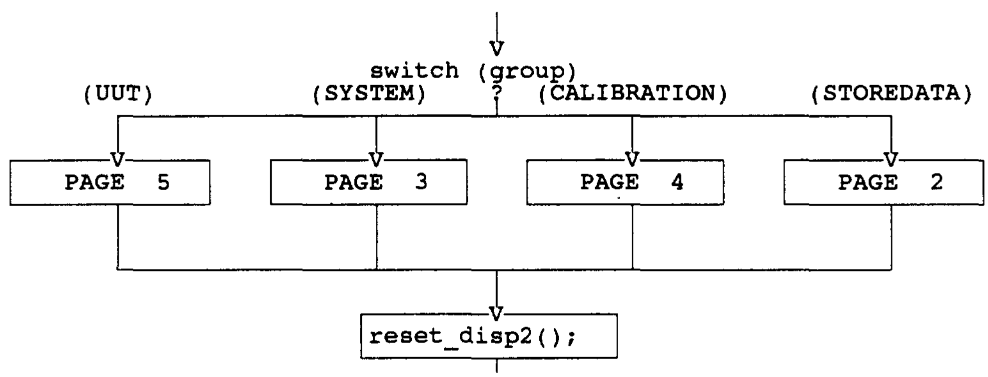

if (password_ok)
?
T
F
if (++element > SYS_LENGTH -1)
?
T
F
element=0;
new_prog_data();
break;

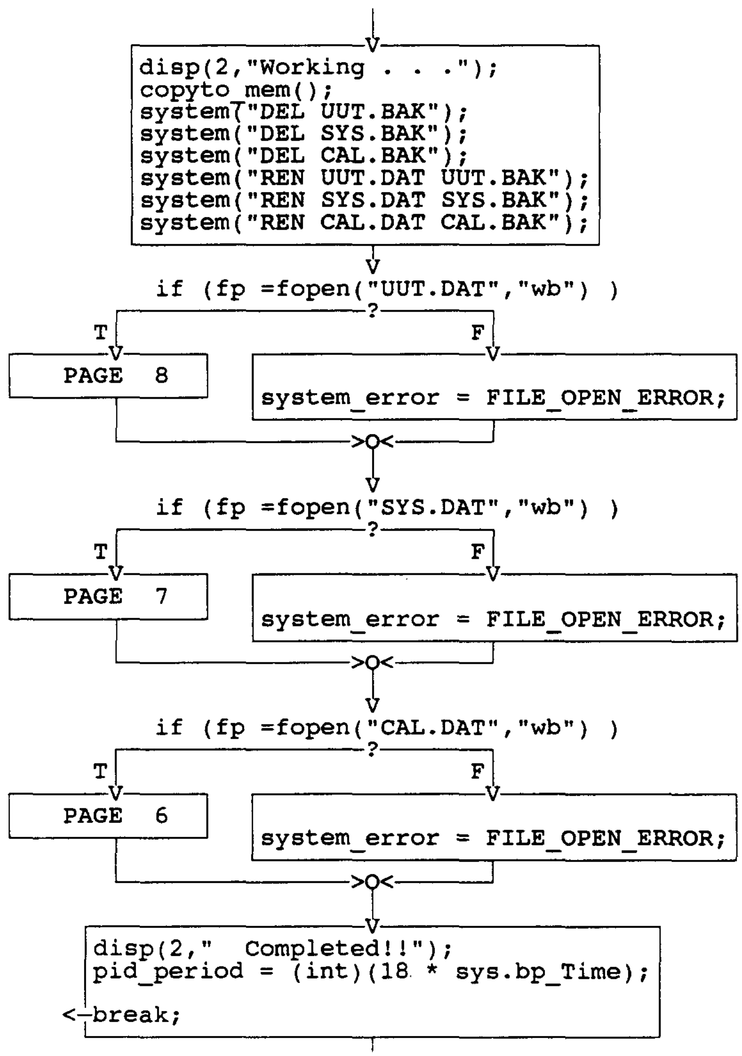
```
disp(1,"CALIBRATION ??");
disp(2," 1=Yes Enter>");
<-break;
```

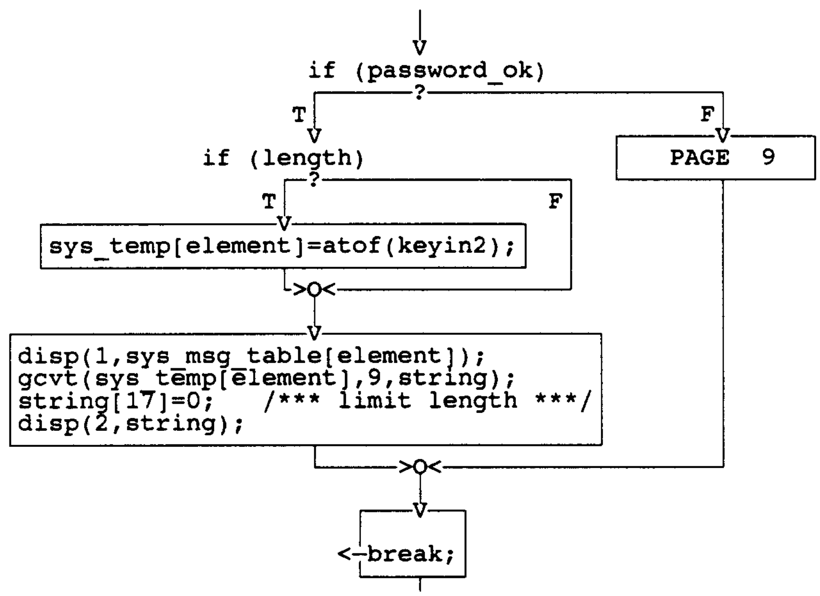
```
disp(1,"SYS Data <ENTER");
disp(2,"PASSWORD>");
<-break;
```

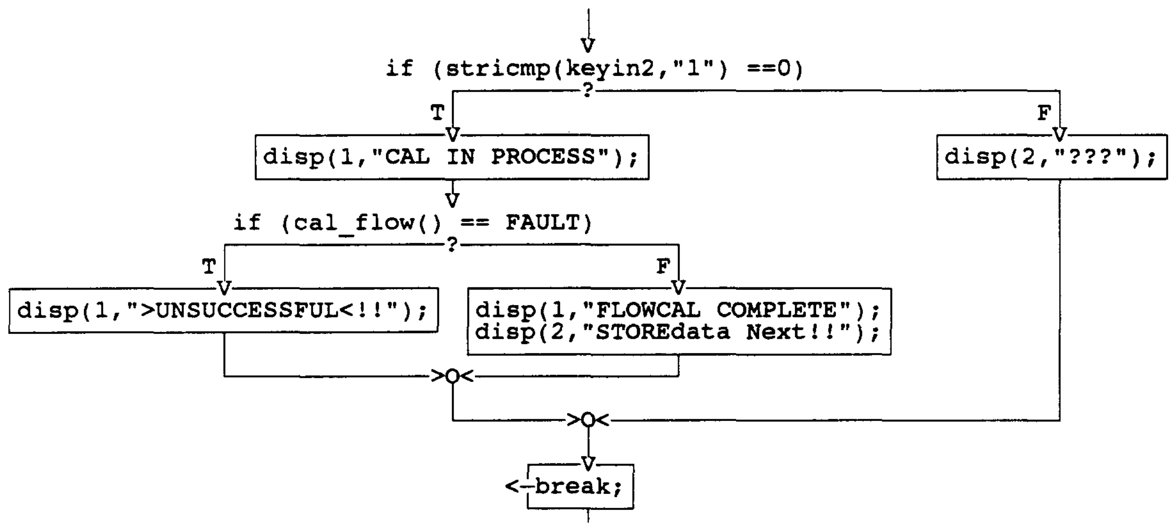

if (charin == UPARROW)
T
F
if ((keyin2[--i]+=1) >'Z'|| (keyin2[i] <'A'))
/*** next..
PAGE 15
T
F
keyin2[i]='A';


if ((keyin2[--i]-=1) <'A')
/*** next...
?
T
F
keyin2[i]='Z';


switch (group)
?
(UUT)
(SYSTEM)
(CALIBRATION)
(STOREDATA)
PAGE 5
PAGE 3
PAGE 4
PAGE 2
reset_disp2();


disp(2,"Working . . .");
copyto_mem();
system("DEL UUT.BAK");
system("DEL SYS.BAK");
system("DEL CAL.BAK");
system("REN UUT.DAT UUT.BAK");
system("REN SYS.DAT SYS.BAK");
system("REN CAL.DAT CAL.BAK");
if (fp =fopen("UUT.DAT","wb"))
T
F
PAGE 8
system_error = FILE_OPEN_ERROR;
if (fp =fopen("SYS.DAT","wb"))
T
F
PAGE 7
system_error = FILE_OPEN_ERROR;
if (fp =fopen("CAL.DAT","wb"))
T
F
PAGE 6
system_error = FILE_OPEN_ERROR;
disp(2," Completed!!");
pid_period = (int)(18 * sys.bp_Time);
break;

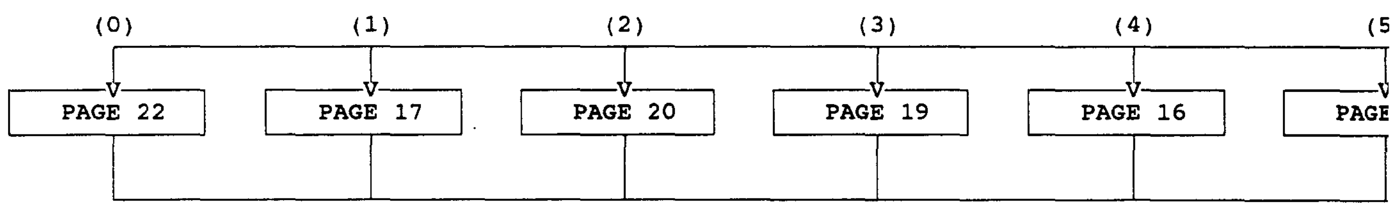

if (password_ok)
?
T
F
PAGE 9
if (length)
?
T
F
sys_temp[element]=atof(keyin2);
disp(1,sys_msg_table[element]);
gcvt(sys_temp[element],9,string);
string[17]=0; /*** limit length ***/
disp(2,string);
<-break;

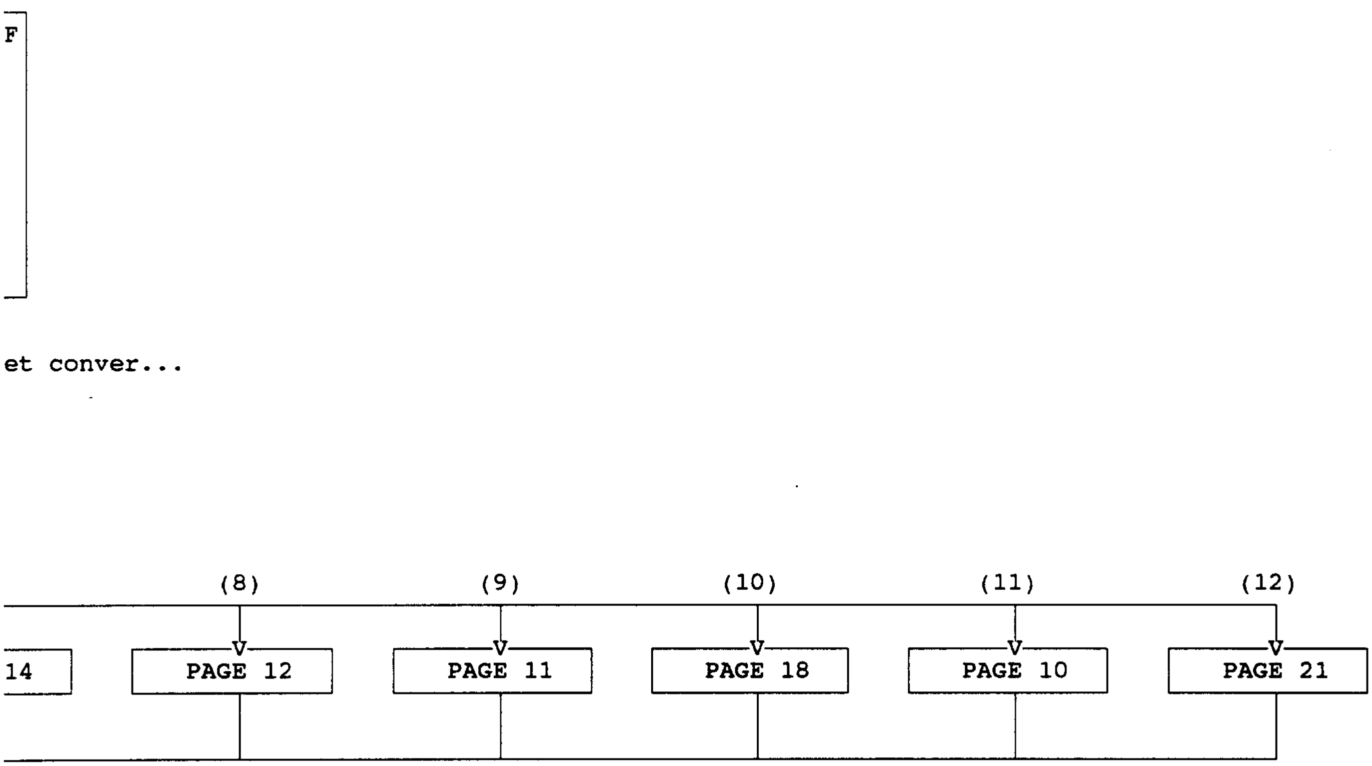

if (stricmp(keyin2,"1") ==0)
?
T
F
disp(1,"CAL IN PROCESS");
disp(2,"???");
if (cal_flow() == FAULT)
?
T
F
disp(1,">UNSUCCESSFUL<!!");
disp(1,"FLOWCAL COMPLETE");
disp(2,"STOREdata Next!!");
<-break;

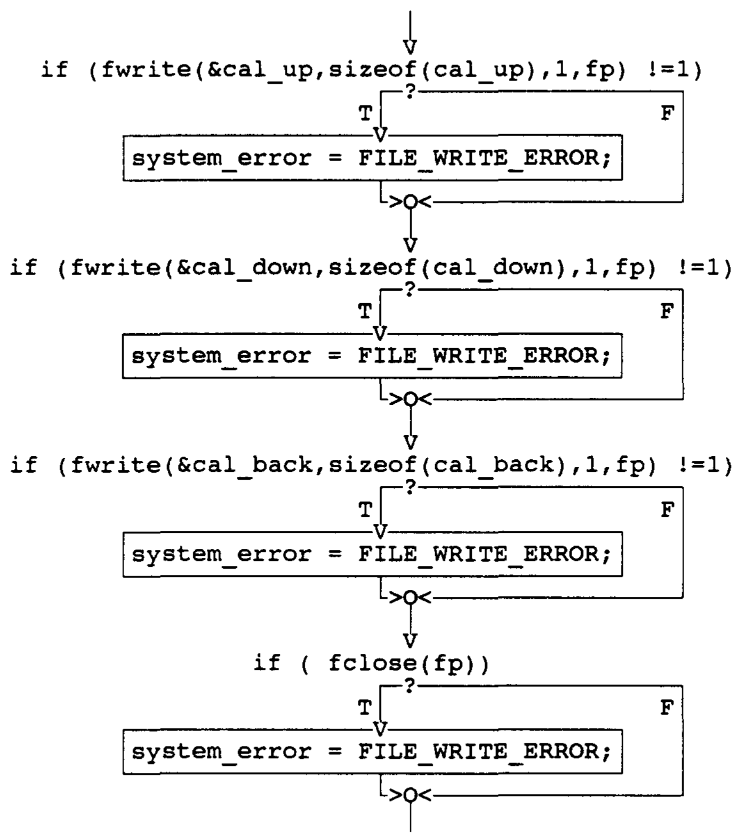
if

<

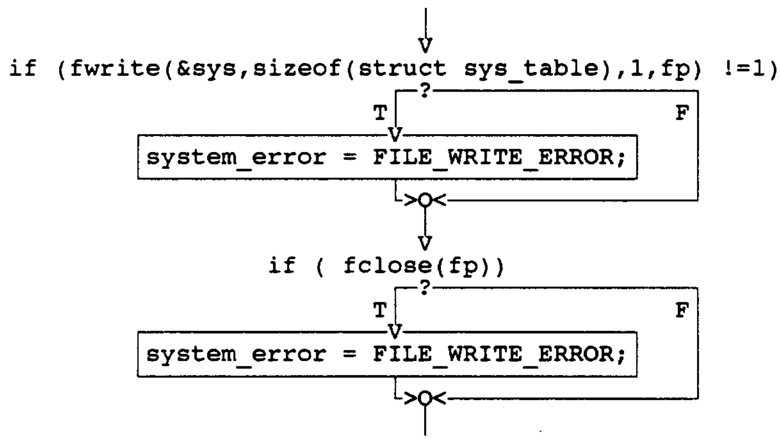
if (element !

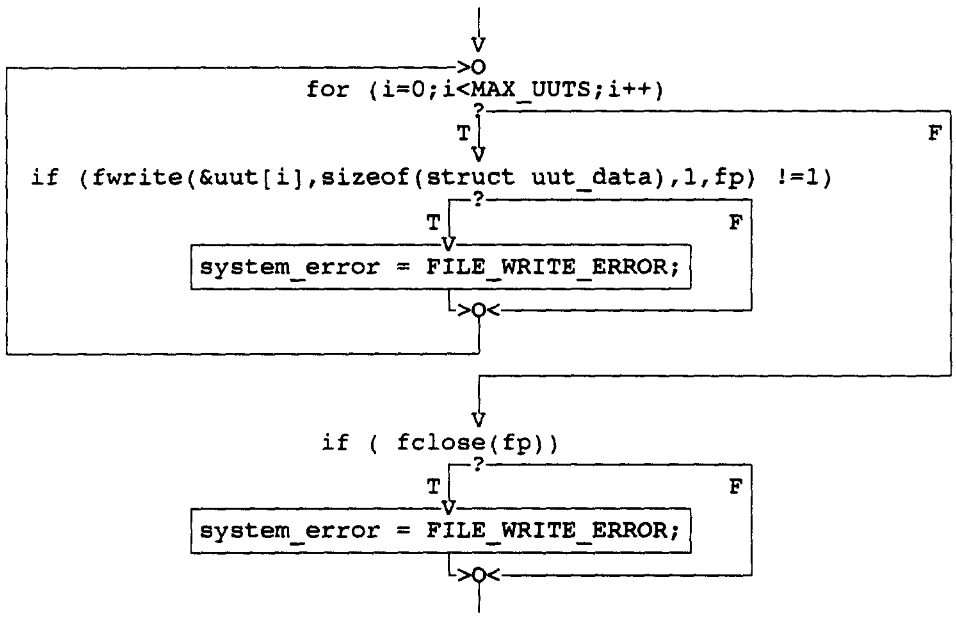


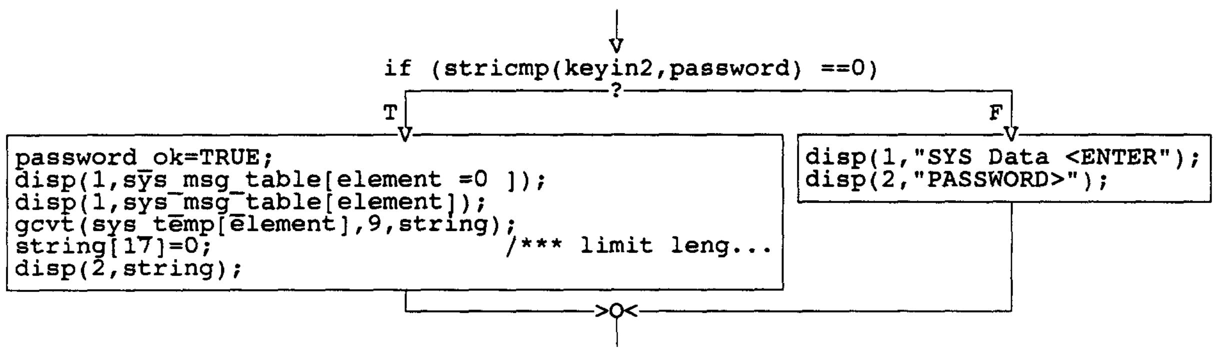

F
et conver...
(8)
(9)
(10)
(11)
(12)
14
PAGE 12
PAGE 11
PAGE 18
PAGE 10
PAGE 21



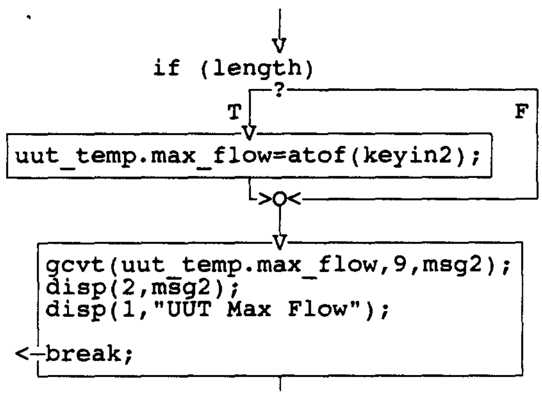
if (fwrite(&cal_up,sizeof(cal_up),1,fp) !=1)
?
T
F
system_error = FILE_WRITE_ERROR;
if (fwrite(&cal_down,sizeof(cal_down),1,fp) !=1)
?
T
F
system_error = FILE_WRITE_ERROR;
if (fwrite(&cal_back,sizeof(cal_back),1,fp) !=1)
?
T
F
system_error = FILE_WRITE_ERROR;
if (fclose(fp))
?
T
F
system_error = FILE_WRITE_ERROR;

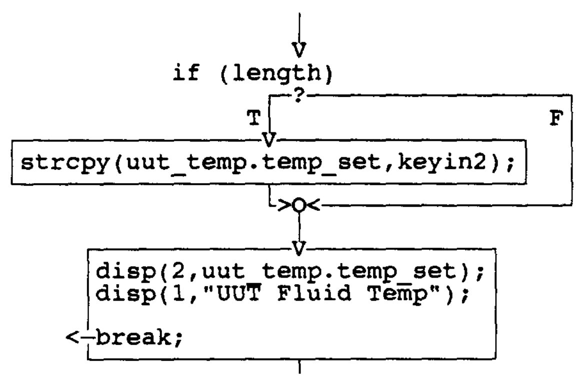

if (fwrite(&sys,sizeof(struct sys_table),1,fp) !=1)
?
T
F
system_error = FILE_WRITE_ERROR;
if (fclose(fp))
?
T
F
system_error = FILE_WRITE_ERROR;

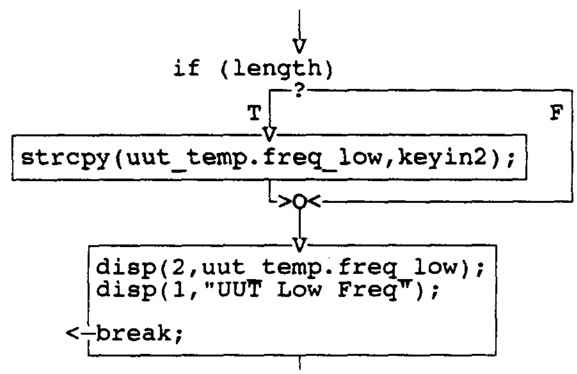

for (i=0;i<MAX_UUTS;i++)
T
F
if (fwrite(&uut[i],sizeof(struct uut_data),1,fp) !=1)
T
F
system_error = FILE_WRITE_ERROR;
if (fclose(fp))
T
F
system_error = FILE_WRITE_ERROR;

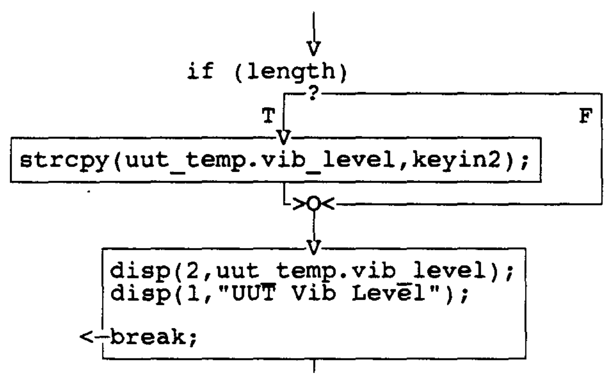

if (stricmp(keyin2,password) ==0)
?
T
F
password_ok=TRUE;
disp(1,sys_msg_table[element =0]);
disp(1,sys_msg_table[element]);
gcvt(sys_temp[element],9,string);
string[17]=0; /*** limit leng...
disp(2,string);
disp(1,"SYS Data <ENTER>");
disp(2,"PASSWORD>");

```
if (length)
    T: uut_temp.ind_type=atoi(keyin2);
    F

itoa(uut_temp.ind_type,msg2,10);
disp(2,msg2);
disp(1,"UUT Indictr Type");

<-break;
```

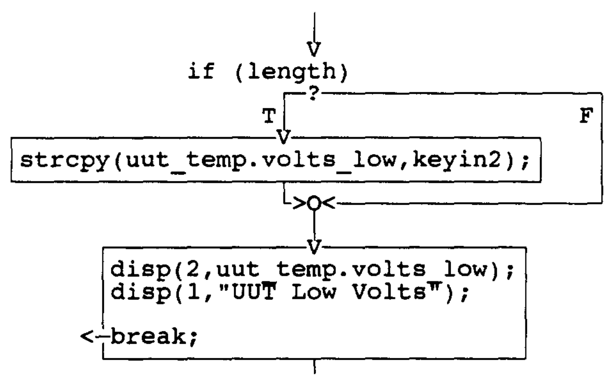

if (length)
?
T
F
uut_temp.max_flow=atof(keyin2);
gcvt(uut_temp.max_flow,9,msg2);
disp(2,msg2);
disp(1,"UUT Max Flow");
break;

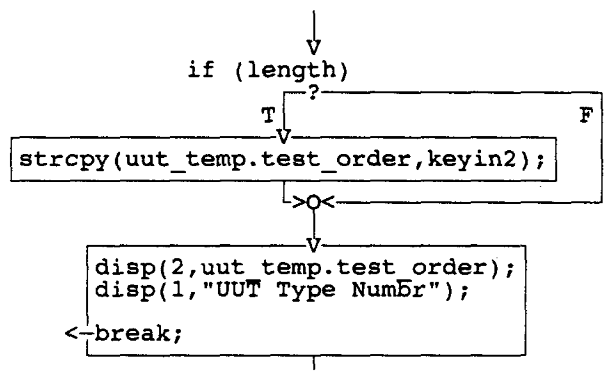

if (length)
?
T
F
strcpy(uut_temp.temp_set,keyin2);
disp(2,uut_temp.temp_set);
disp(1,"UUT Fluid Temp");
break;

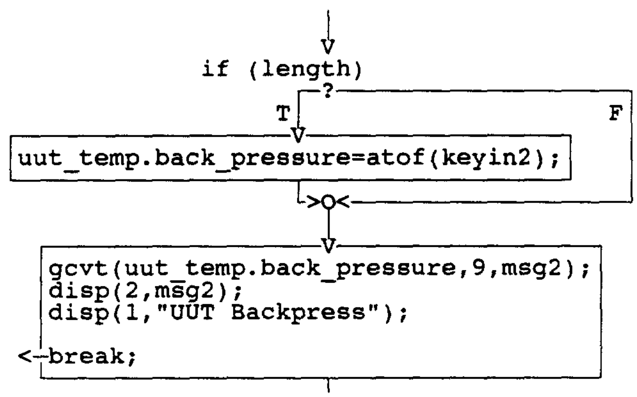

if (length)
?
T
F
strcpy(uut_temp.freq_low,keyin2);
disp(2,uut_temp.freq_low);
disp(1,"UUT Low Freq");
break;


if (length)
?
T
F
strcpy(uut_temp.vib_level,keyin2);
disp(2,uut_temp.vib_level);
disp(1,"UUT Vib Level");
break;

```
if (length)
?
T                                         F
strcpy(uut_temp.phase_low,keyin2);

disp(2,uut_temp.phase_low);
disp(1,"UUT Low Phase");

<-break;
```

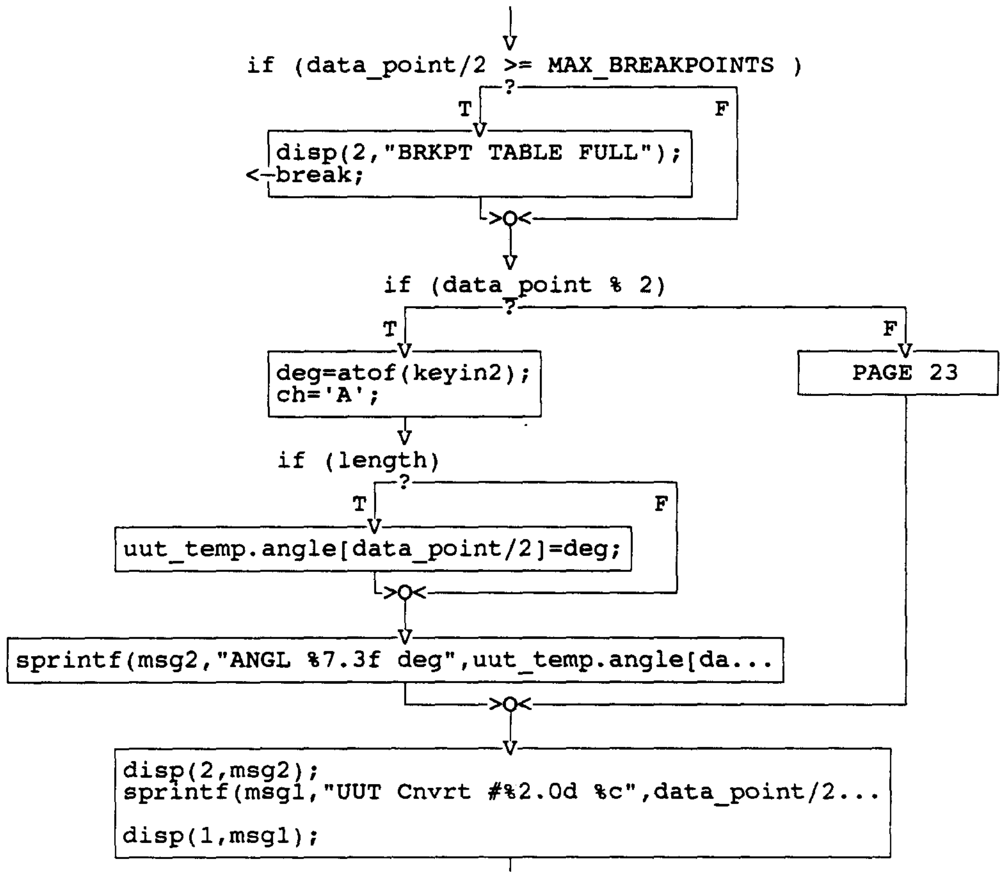

if (length)
?
T
F
strcpy(uut_temp.volts_low,keyin2);
disp(2,uut_temp.volts_low);
disp(1,"UUT Low Volts");
break;

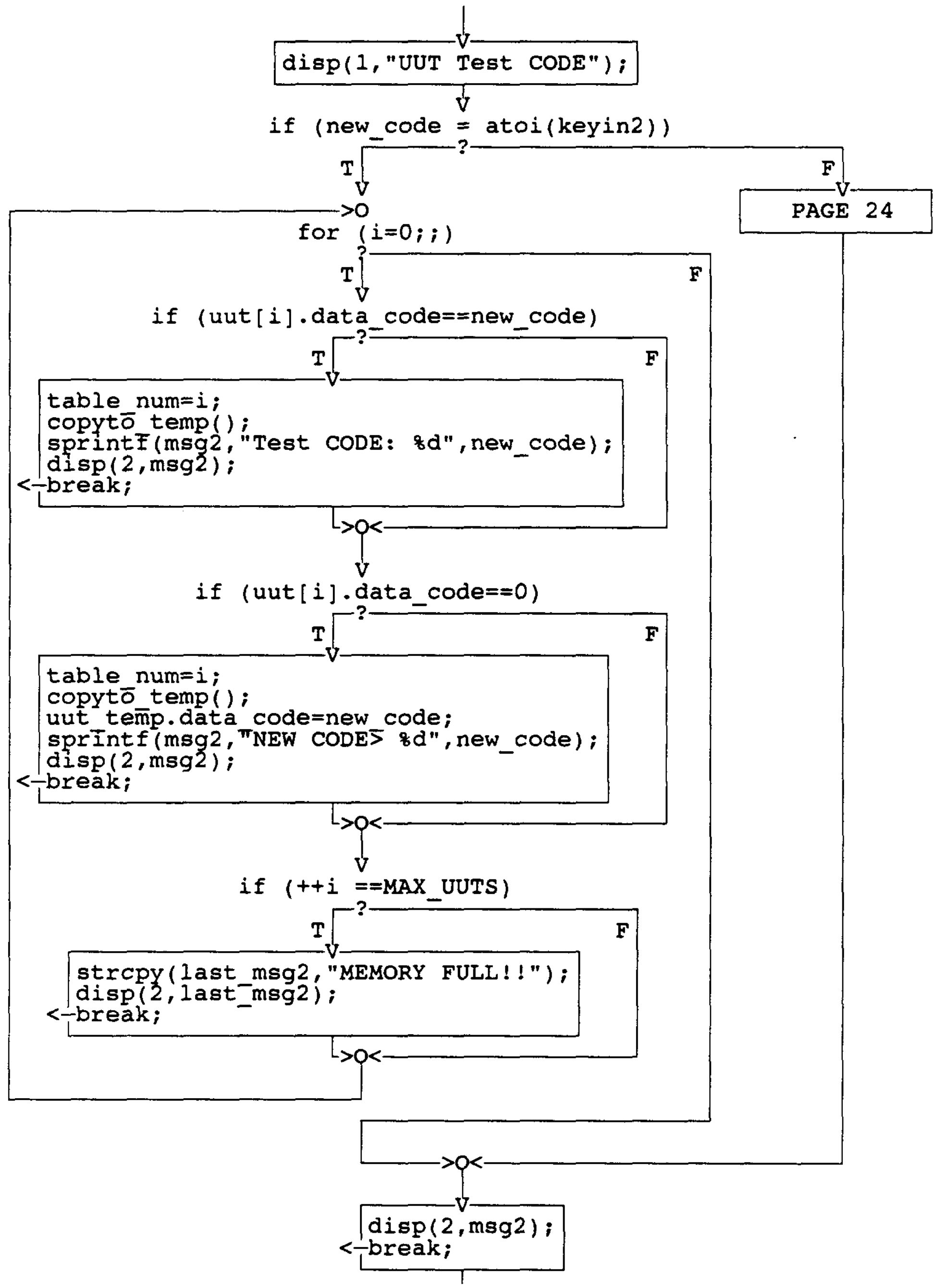

if (length)
?
T
F
strcpy(uut_temp.test_order,keyin2);
disp(2,uut_temp.test_order);
disp(1,"UUT Type Numbr");
break;

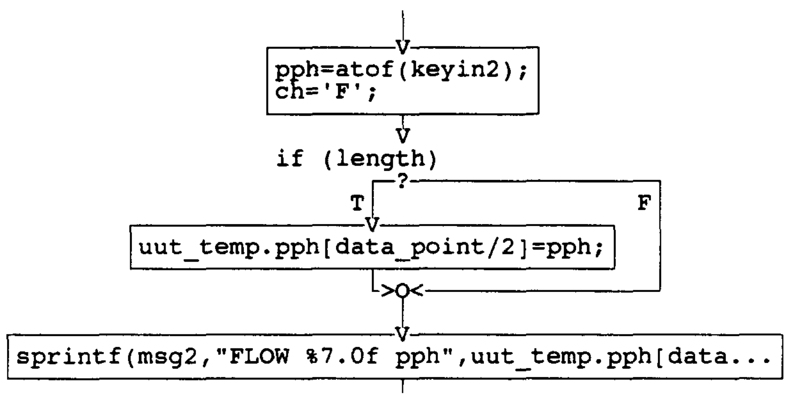

if (length)
?
T
F
uut_temp.back_pressure=atof(keyin2);
gcvt(uut_temp.back_pressure,9,msg2);
disp(2,msg2);
disp(1,"UUT Backpress");
break;

```
if (length)
  T  ?  F
strcpy(uut_temp.volts_400hz,keyin2);

disp(2,uut_temp.volts_400hz);
disp(1,"UUT High Volts");

<-break;
```

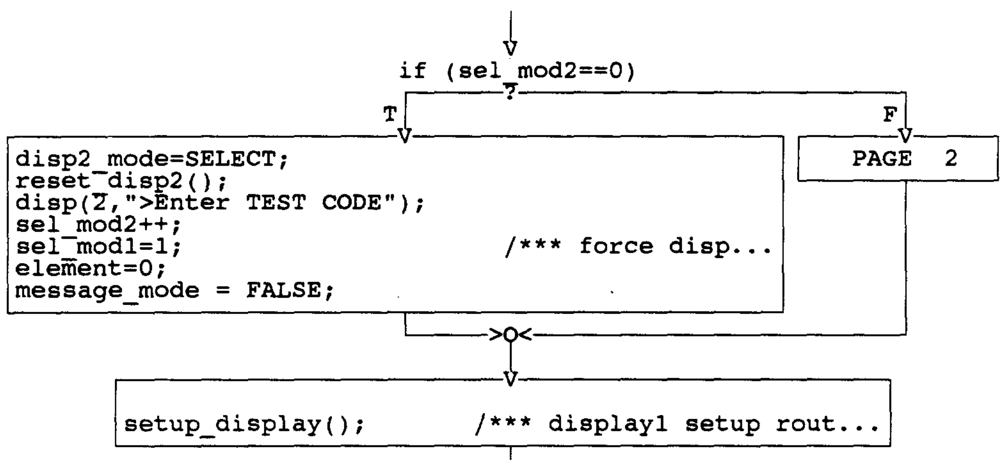
```
if (length)
T                                   F
strcpy(uut_temp.adaptor_num,keyin2);

disp(2,uut_temp.adaptor_num);
disp(1,"UUT Adaptor Cable");

<-break;
```

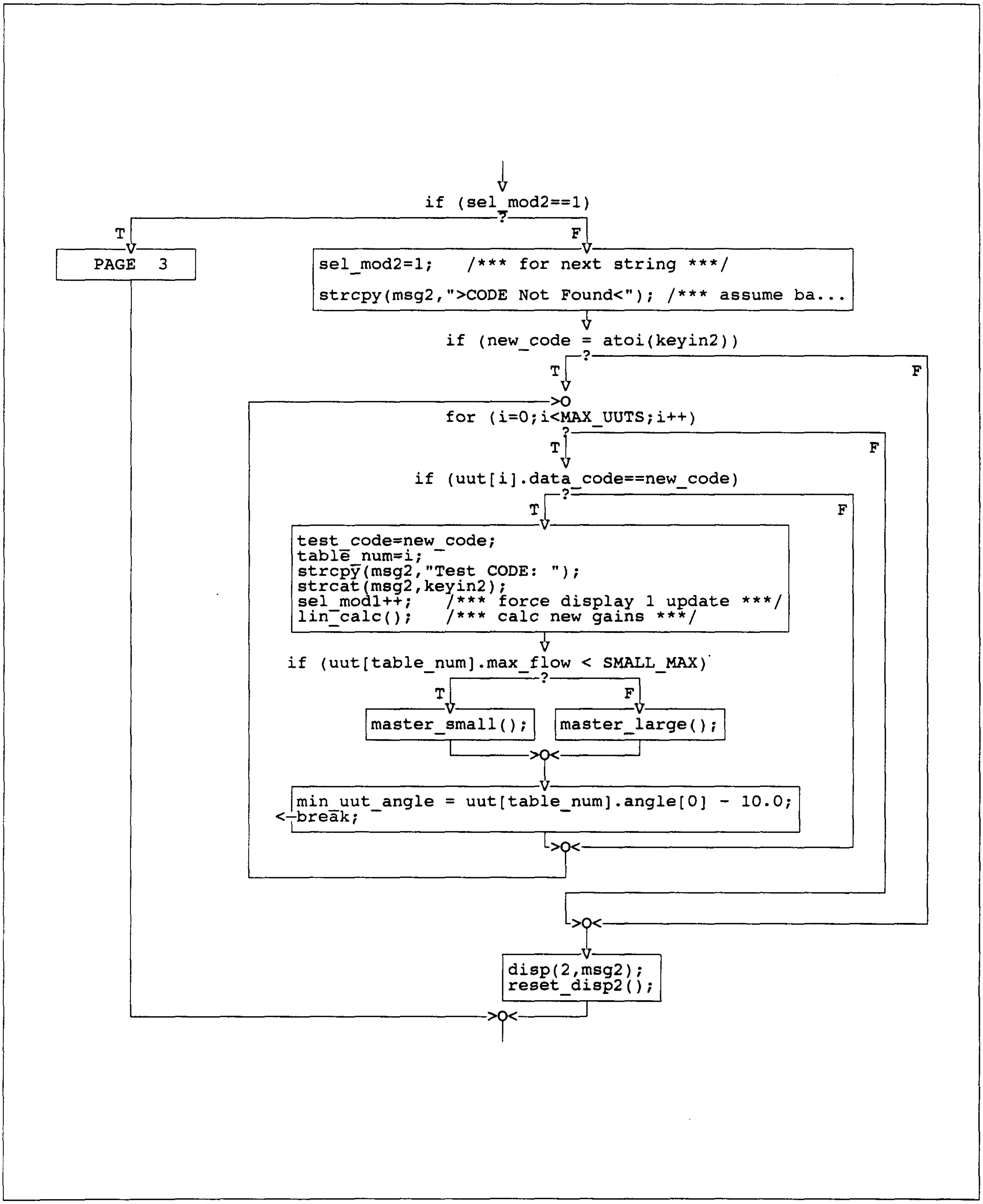

if (data_point/2 >= MAX_BREAKPOINTS)
?
T
F
disp(2,"BRKPT TABLE FULL");
break;
if (data_point % 2)
?
T
F
deg=atof(keyin2);
ch='A';
PAGE 23
if (length)
?
T
F
uut_temp.angle[data_point/2]=deg;
sprintf(msg2,"ANGL %7.3f deg",uut_temp.angle[da...
disp(2,msg2);
sprintf(msg1,"UUT Cnvrt #%2.0d %c",data_point/2...
disp(1,msg1);

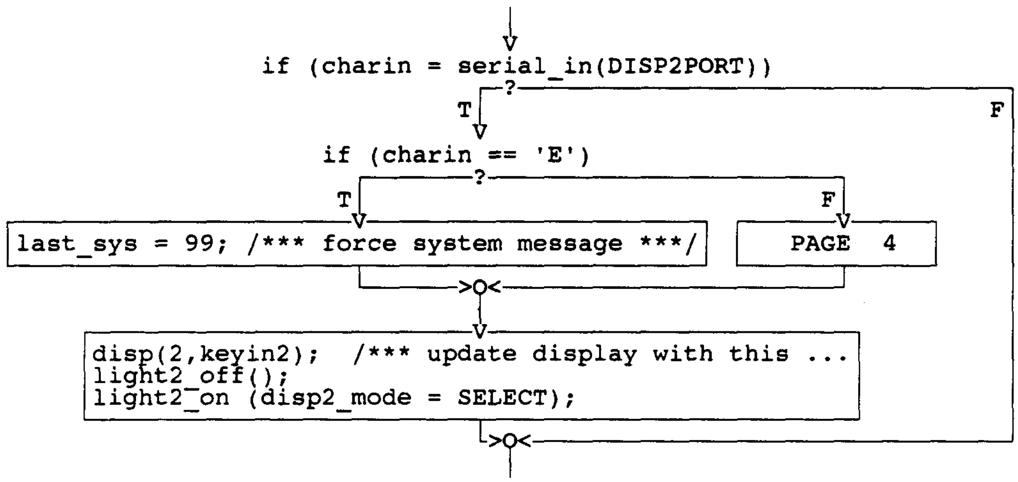

disp(1,"UUT Test CODE");
if (new_code = atoi(keyin2))
?
T
F
PAGE 24
for (i=0;;)
?
T
F
if (uut[i].data_code==new_code)
?
T
F
table_num=i;
copyto_temp();
sprintf(msg2,"Test CODE: %d",new_code);
disp(2,msg2);
break;
if (uut[i].data_code==0)
?
T
F
table_num=i;
copyto_temp();
uut_temp.data_code=new_code;
sprintf(msg2,"NEW CODE> %d",new_code);
disp(2,msg2);
break;
if (++i ==MAX_UUTS)
?
T
F
strcpy(last_msg2,"MEMORY FULL!!");
disp(2,last_msg2);
break;
disp(2,msg2);
break;


pph=atof(keyin2);
ch='F';
if (length)
?
T
F
uut_temp.pph[data_point/2]=pph;
sprintf(msg2,"FLOW %7.0f pph",uut_temp.pph[data...

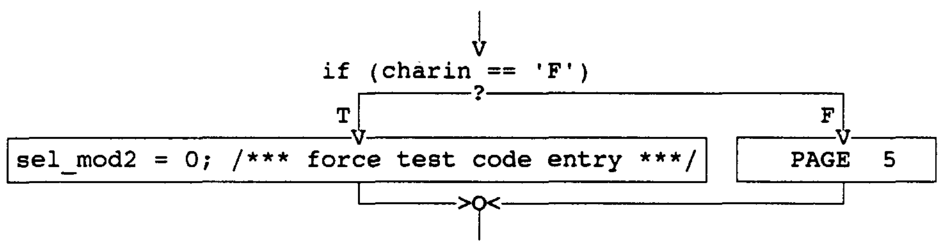
```
sprintf(msg2,"Enter New> %d",uut[table_num].dat...
```

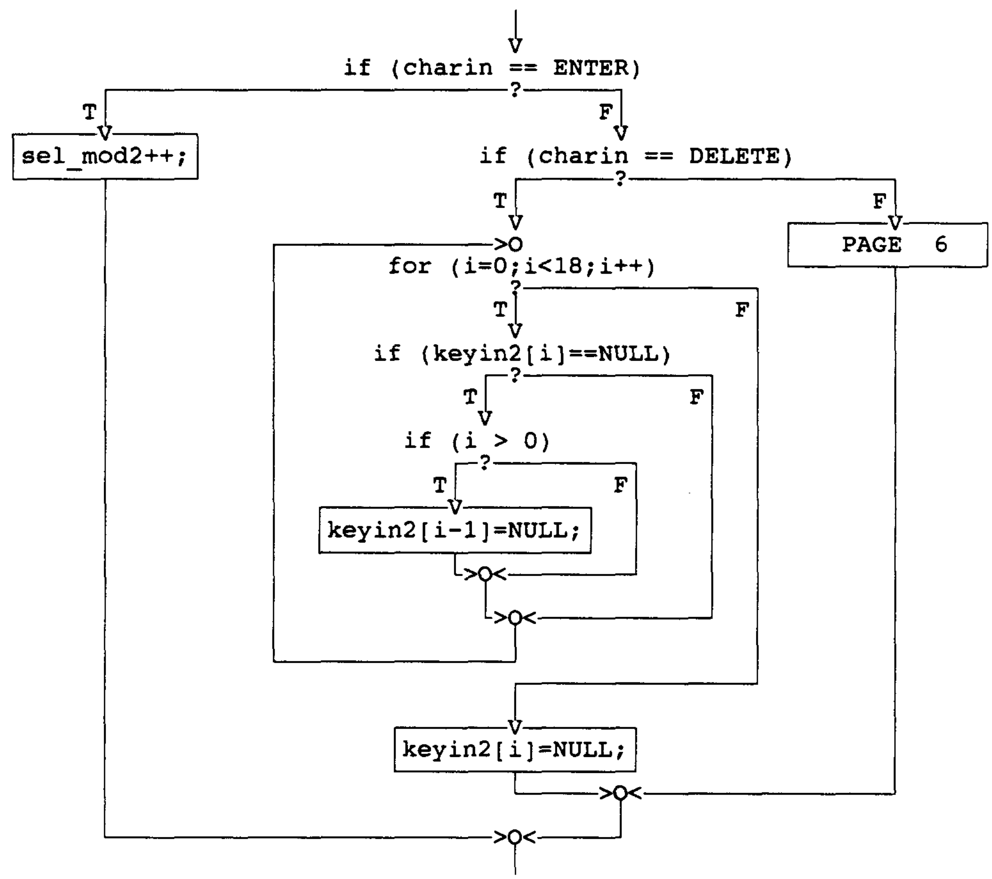

if (sel_mod2==0)
T
F
disp2_mode=SELECT;
reset_disp2();
disp(2,">Enter TEST CODE");
sel_mod2++;
sel_mod1=1;
/*** force disp...
element=0;
message_mode = FALSE;
PAGE 2
setup_display();
/*** display1 setup rout...


if (sel_mod2==1)
T
F
PAGE 3
sel_mod2=1; /*** for next string ***/
strcpy(msg2,">CODE Not Found<"); /*** assume ba...
if (new_code = atoi(keyin2))
T
F
for (i=0;i<MAX_UUTS;i++)
T
F
if (uut[i].data_code==new_code)
T
F
test_code=new_code;
table_num=i;
strcpy(msg2,"Test CODE: ");
strcat(msg2,keyin2);
sel_mod1++; /*** force display 1 update ***/
lin_calc(); /*** calc new gains ***/
if (uut[table_num].max_flow < SMALL_MAX)
T
F
master_small();
master_large();
min_uut_angle = uut[table_num].angle[0] - 10.0;
break;
disp(2,msg2);
reset_disp2();



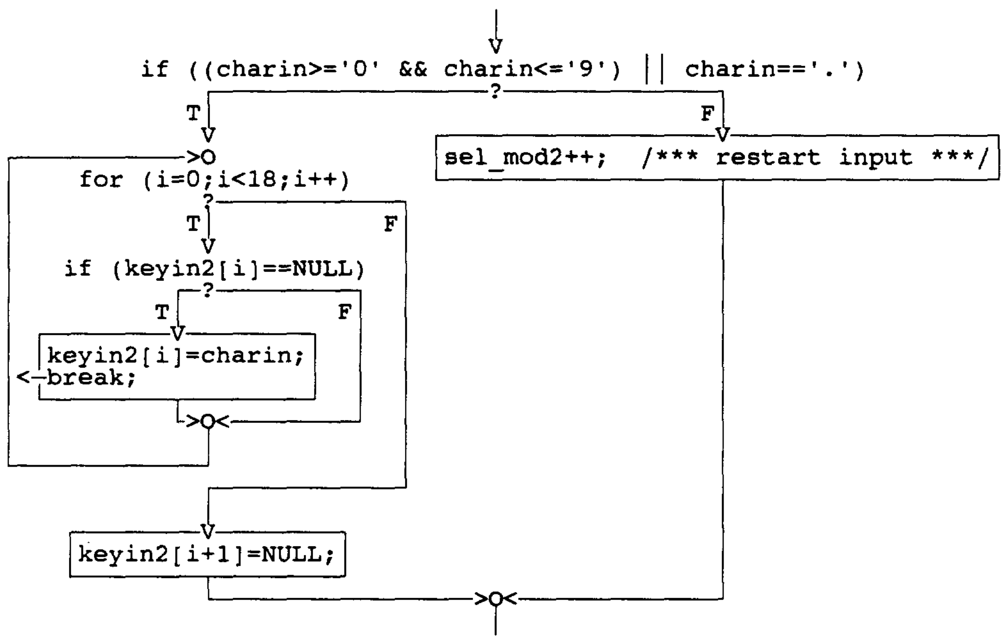
if (charin = serial_in(DISP2PORT))
?
T
F
if (charin == 'E')
?
T
F
last_sys = 99; /*** force system message ***/
PAGE 4
disp(2,keyin2); /*** update display with this ...
light2_off();
light2_on (disp2_mode = SELECT);

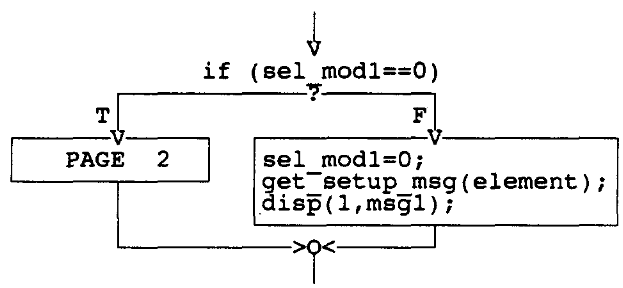

if (charin == 'F')
?
T
F
sel_mod2 = 0; /*** force test code entry ***/
PAGE 5

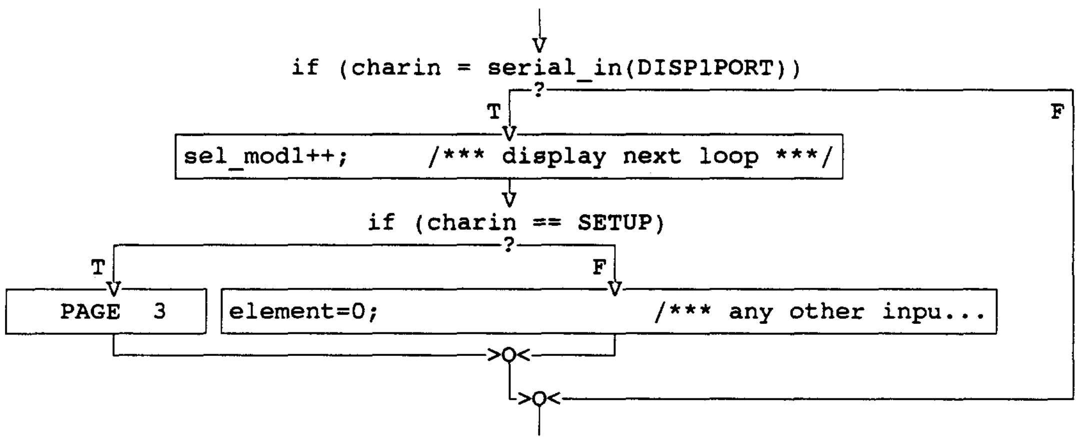


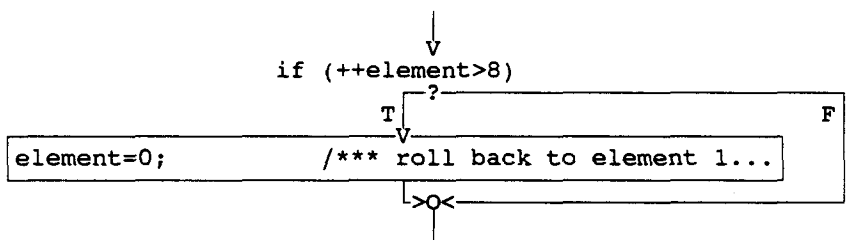
if (charin == ENTER)
?
T
F
sel_mod2++;
if (charin == DELETE)
?
T
F
PAGE 6
for (i=0;i<18;i++)
?
T
F
if (keyin2[i]==NULL)
?
T
F
if (i > 0)
?
T
F
keyin2[i-1]=NULL;
keyin2[i]=NULL;

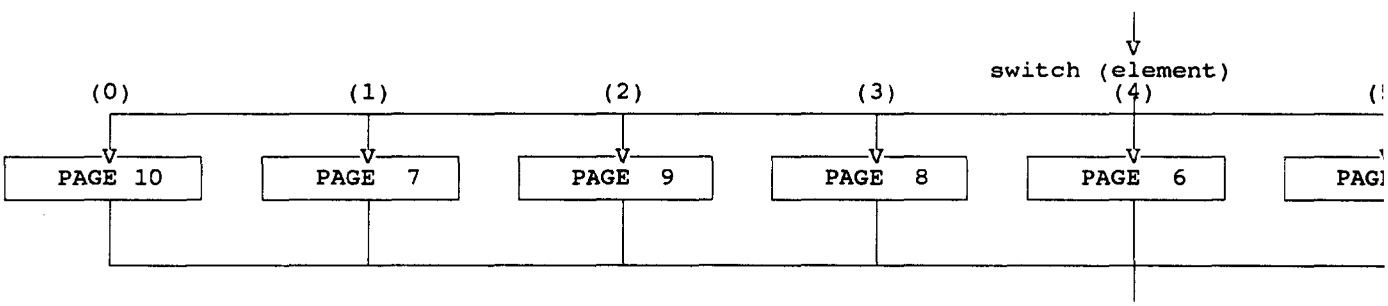
```
if ((charin == CLEAR) & message_mode)
?
T
strcpy (sys_message,"  SYSTEM OK");
gpib_fault = 0;
last_sys = 99; /*** force system message update...
horn = OFF;
F
PAGE  7
```

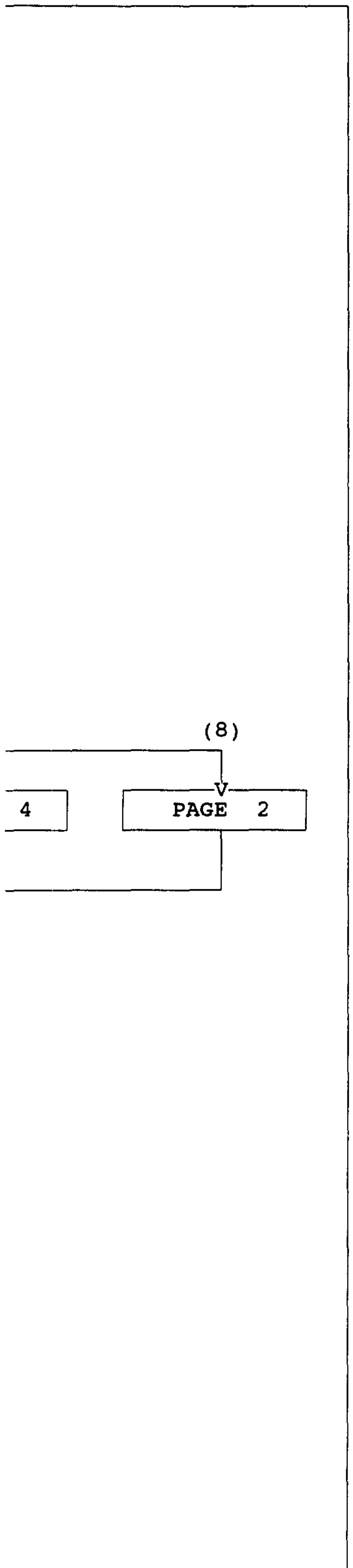

if ((charin>='0' && charin<='9') || charin=='.')
T
F
sel_mod2++; /*** restart input ***/
for (i=0;i<18;i++)
T
F
if (keyin2[i]==NULL)
T
F
keyin2[i]=charin;
break;
keyin2[i+1]=NULL;


if (sel_mod1==0)
T
F
PAGE 2
sel_mod1=0;
get_setup_msg(element);
disp(1,msg1);


if (charin = serial_in(DISP1PORT))
?
T
F
sel_modl++;   /*** display next loop ***/
if (charin == SETUP)
?
T
F
PAGE 3
element=0;   /*** any other inpu...


if (++element>8)
?
T
F
element=0;
/*** roll back to element 1...


switch (element)
(0)
(1)
(2)
(3)
(4)
PAGE 10
PAGE 7
PAGE 9
PAGE 8
PAGE 6


(8)
4
PAGE 2

```
strcpy(msg1,"TEMP Set: ");
strcat(msg1,uut[table_num].temp_set);
<-break;
```

```
strcpy(msg1,"Low FREQ: ");
strcat(msg1,uut[table_num].freq_low);
<-break;
```

```
strcpy(msg1,"VIB Lev: ");
strcat(msg1,uut[table_num].vib_level);
<-break;
```

```
strcpy(msg1,"Low PHASE: ");
strcat(msg1,uut[table_num].phase_low);
break;
```

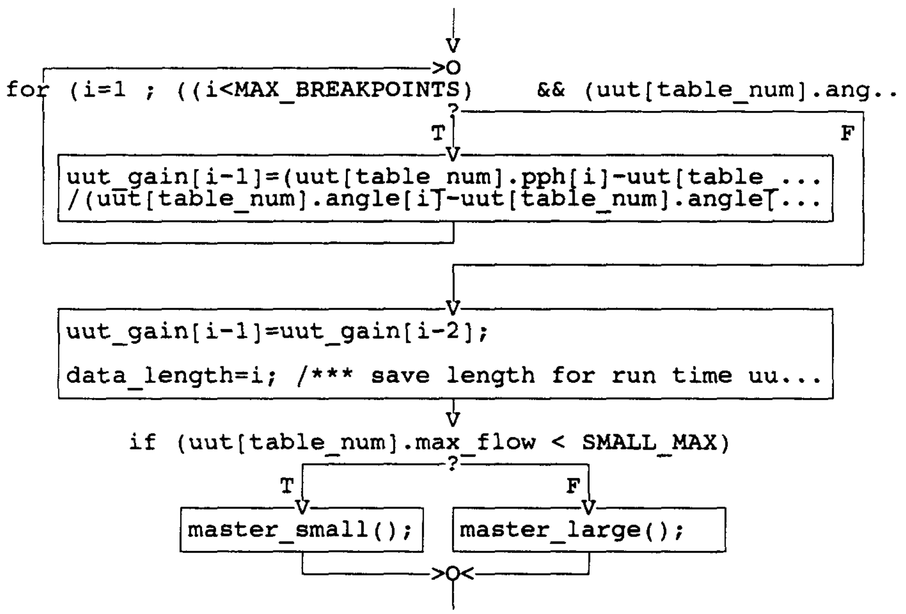
```
strcpy(msg1,"Low Volts: ");
strcat(msg1,uut[table_num].volts_low);
break;
```

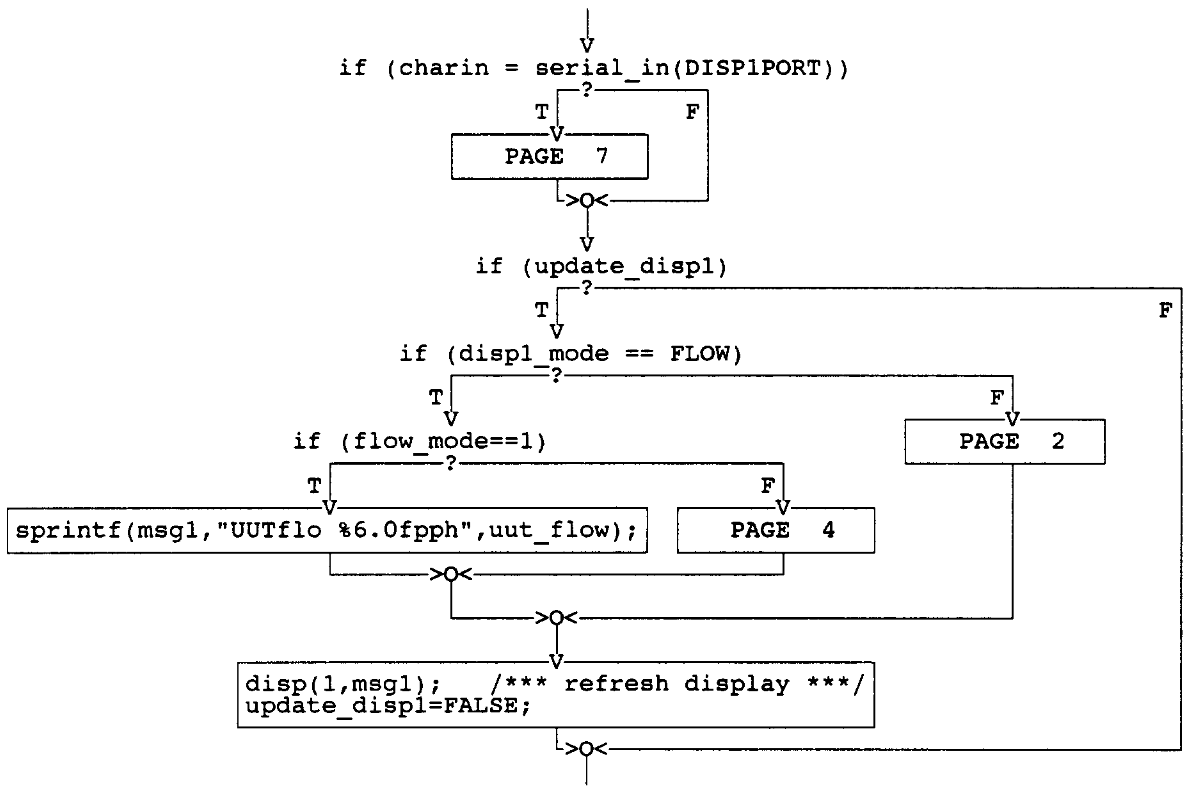
```
strcpy(msg1,"UUT: ");
strcat(msg1,uut[table_num].test_order);
<-break;
```

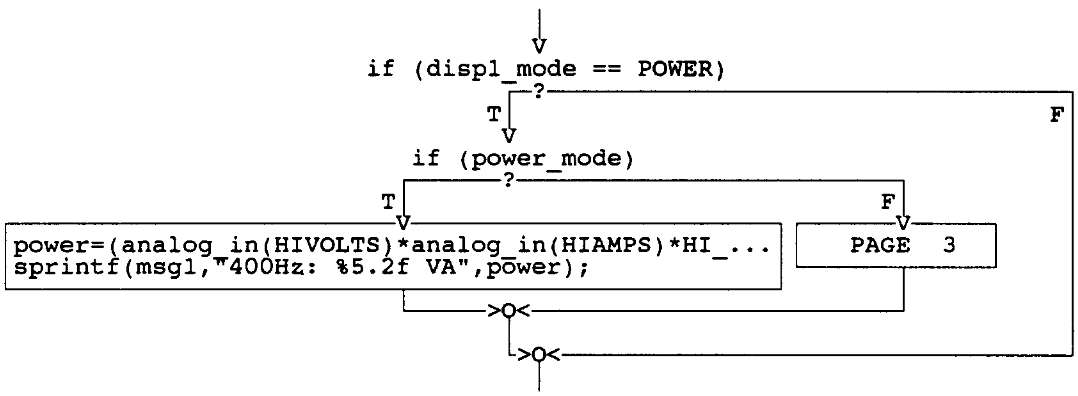
```
strcpy(msg1,"400Hz V: ");
strcat(msg1,uut[table_num].volts_400hz);
break;
```

```
strcpy(msg1,"ADPTR: ");
strcat(msg1,uut[table_num].adaptor_num);
break;
```

```
strcpy(msg1,"Test CODE: ");
itoa(uut[table_num].data_code,string,10);
strcat(msg1,string);
break;
```

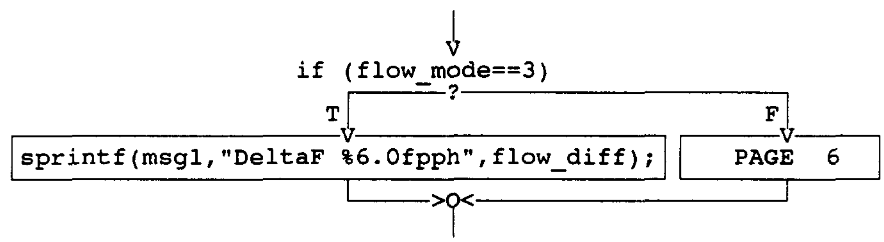

for (i=1 ; ((i<MAX_BREAKPOINTS) && (uut[table_num].ang...
T
F
uut_gain[i-1]=(uut[table_num].pph[i]-uut[table ...
/(uut[table_num].angle[i]-uut[table_num].angle[...
uut_gain[i-1]=uut_gain[i-2];
data_length=i; /*** save length for run time uu...
if (uut[table_num].max_flow < SMALL_MAX)
T
F
master_small();
master_large();


if (charin = serial_in(DISP1PORT))
T
F
PAGE 7
if (update_disp1)
T
F
if (disp1_mode == FLOW)
T
F
PAGE 2
if (flow_mode==1)
T
F
sprintf(msg1,"UUTflo %6.0fpph",uut_flow);
PAGE 4
disp(1,msg1); /*** refresh display ***/
update_disp1=FALSE;

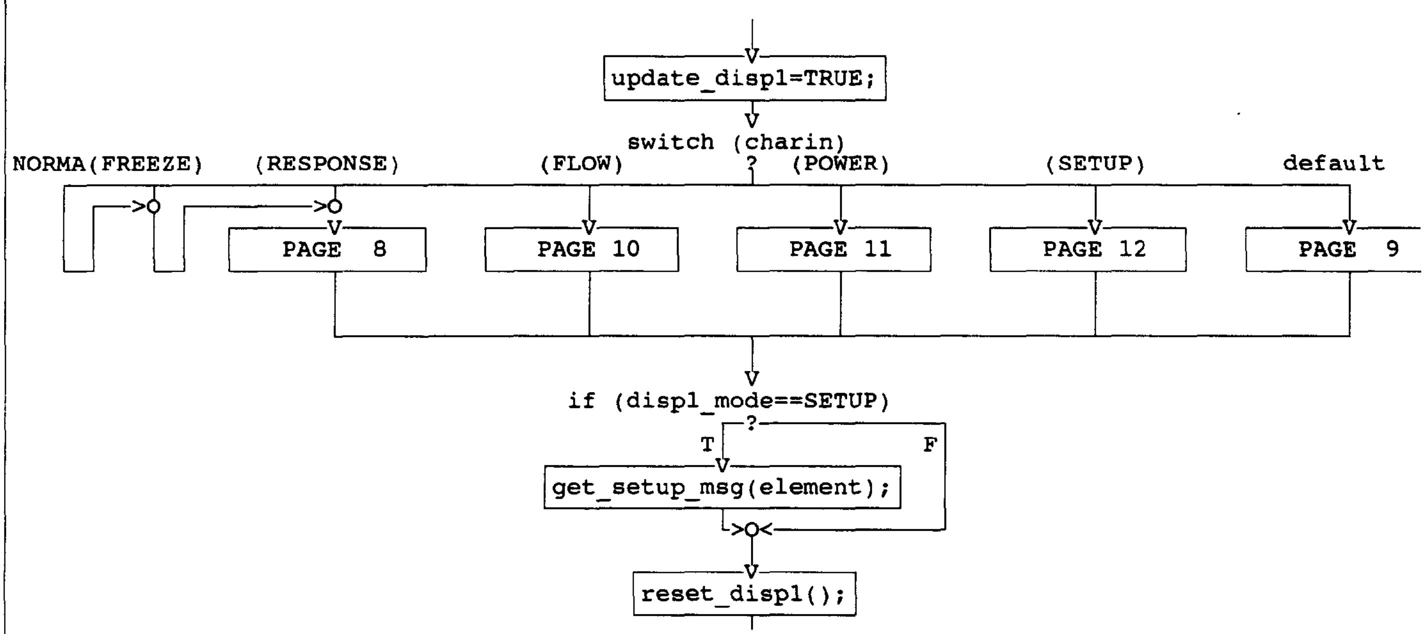

if (displ_mode == POWER)
T
F
if (power_mode)
T
F
power=(analog_in(HIVOLTS)*analog_in(HIAMPS)*HI_...
sprintf(msgl,"400Hz: %5.2f VA",power);
PAGE 3

```
power=(analog_in(LOWVOLTS)*analog_in(LOWAMPS)*L...
sprintf(msg1,"LowV:  %5.2f VA",power);
```

```
                     |
                     V
           if (flow_mode==2)
            ----------?------------------
          T |                           | F
            V                           V
sprintf(msg1,"MASTER %6.0fpph",sys_flow);    PAGE  5
            |                           |
            ----------->O<---------------
                        |
```

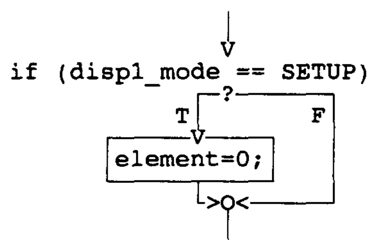

if (flow_mode==3)
?
T
F
sprintf(msg1,"DeltaF %6.0fpph",flow_diff);
PAGE 6

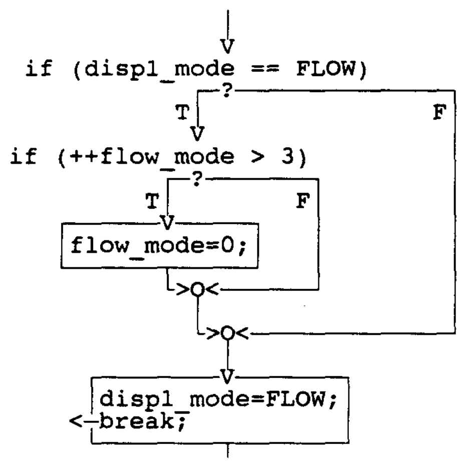
```
sprintf(msg1," Angle %5.2fdeg",uut_angle);
```

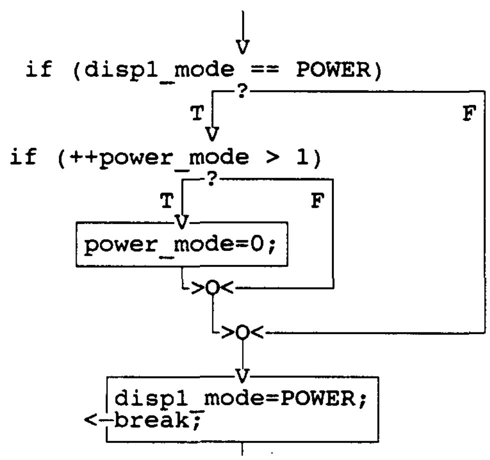


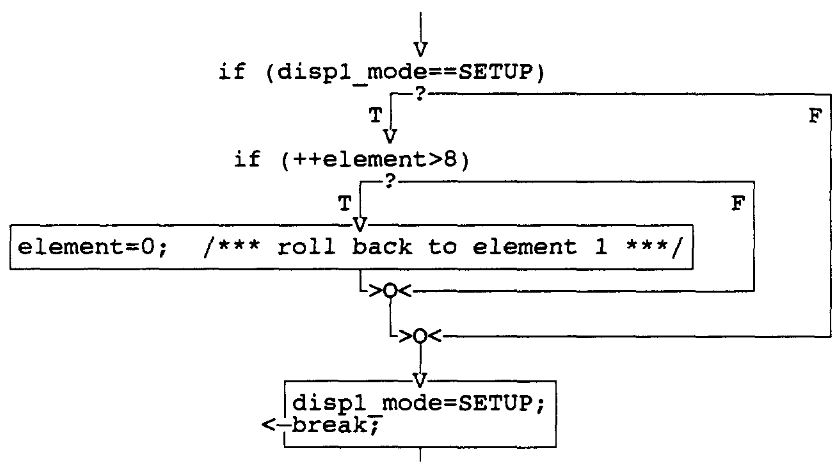
update_displ=TRUE;
switch (charin)
NORMA(FREEZE)
(RESPONSE)
(FLOW)
(POWER)
(SETUP)
default
PAGE 8
PAGE 10
PAGE 11
PAGE 12
PAGE 9
if (displ_mode==SETUP)
T
F
get_setup_msg(element);
reset_displ();

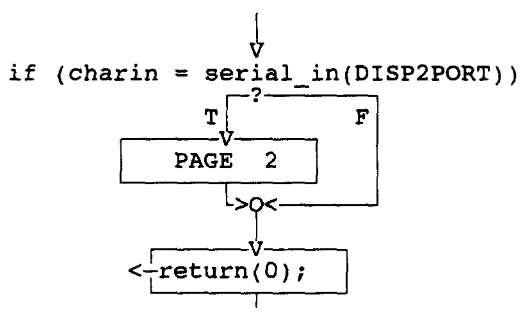
```
control_mode=charin;
<-break;
```

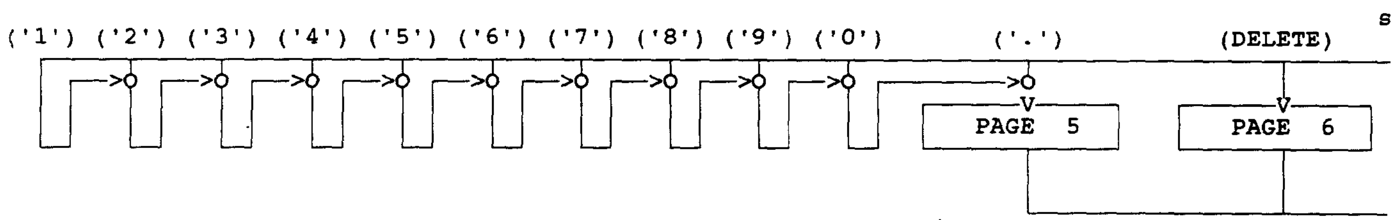

if (displ_mode == SETUP)
?
T
F
element=0;

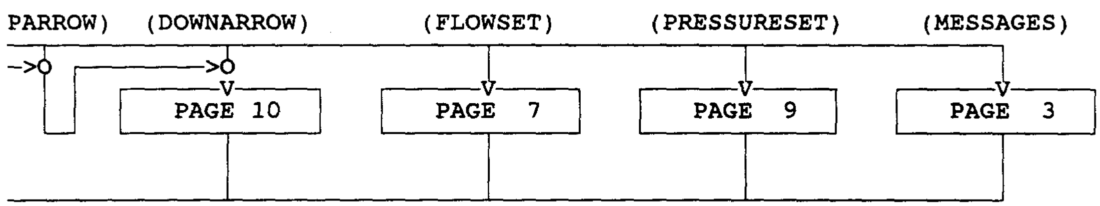

if (displ_mode == FLOW)
?
T
F
if (++flow_mode > 3)
?
T
F
flow_mode=0;
displ_mode=FLOW;
break;


if (displ_mode == POWER)
?
T
F
if (++power_mode > 1)
?
T
F
power_mode=0;
displ_mode=POWER;
break;


if (displ_mode==SETUP)
?
T
F
if (++element>8)
?
T
F
element=0; /*** roll back to element 1 ***/
displ_mode=SETUP;
break;

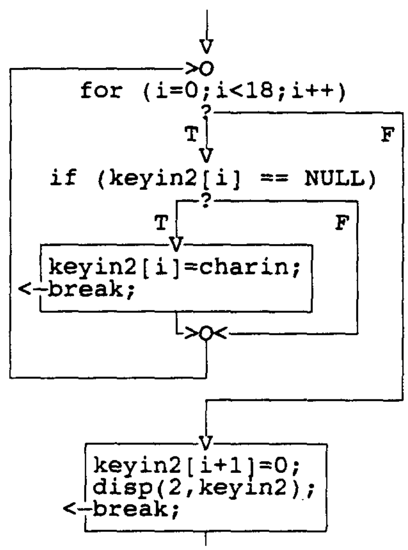

if (charin = serial_in(DISP2PORT))
?
T
F
PAGE 2
return(0);

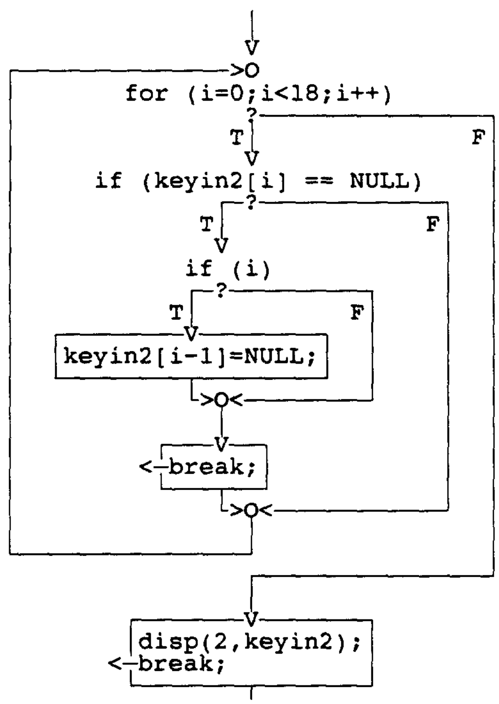

('1') ('2') ('3') ('4') ('5') ('6') ('7') ('8') ('9') ('0')
('.')
(DELETE)
s
PAGE 5
PAGE 6


PARROW)
(DOWNARROW)
(FLOWSET)
(PRESSURESET)
(MESSAGES)
PAGE 10
PAGE 7
PAGE 9
PAGE 3

```
                    |
                    V
disp2_mode = MESSAGES;
reset_disp2();
disp(2,sys_message);

<-break;
```

```
newval = atof(keyin2);
/*** no break; ***/
```

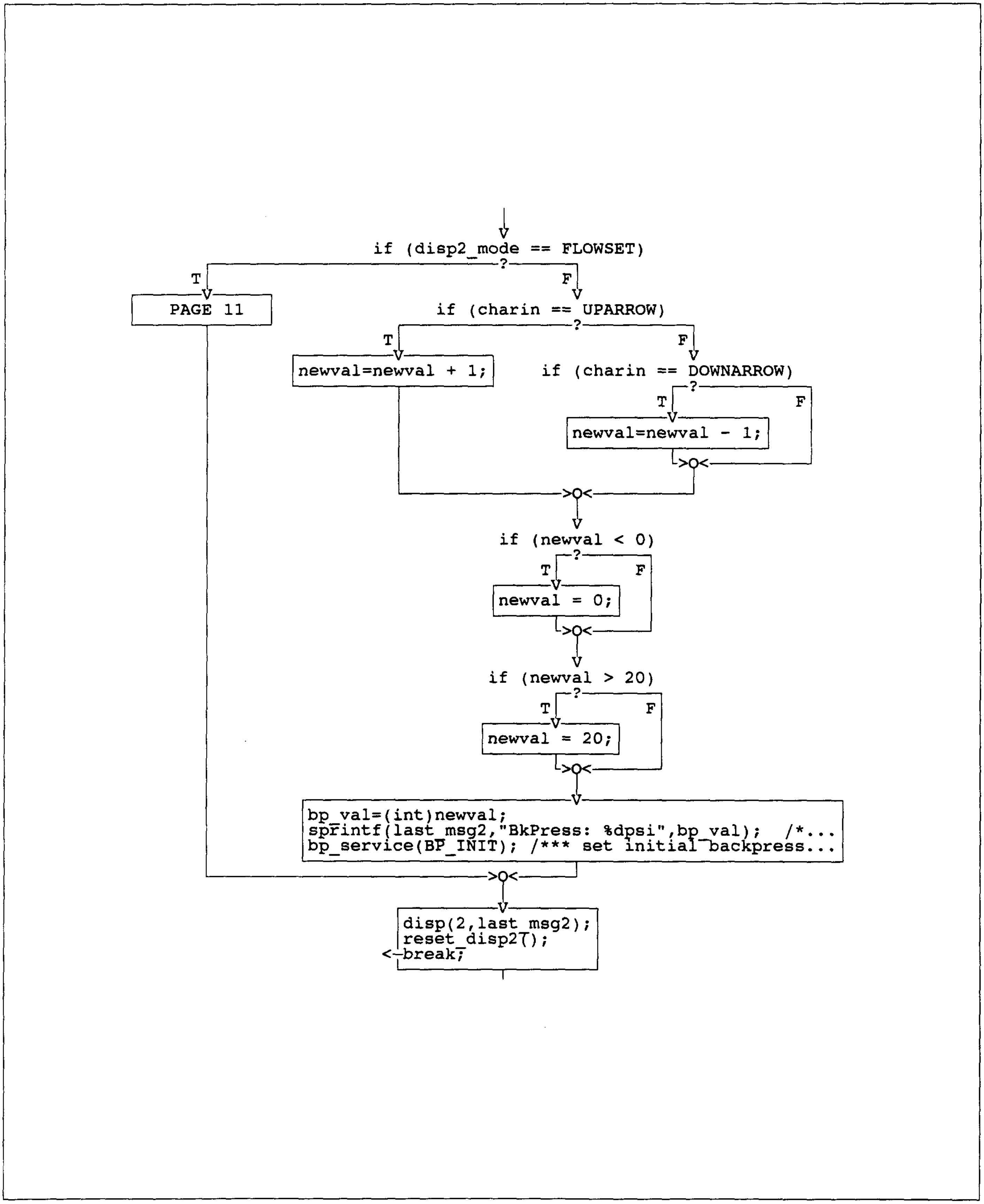

for (i=0;i<18;i++)
T
F
if (keyin2[i] == NULL)
T
F
keyin2[i]=charin;
<-break;
keyin2[i+1]=0;
disp(2,keyin2);
<-break;

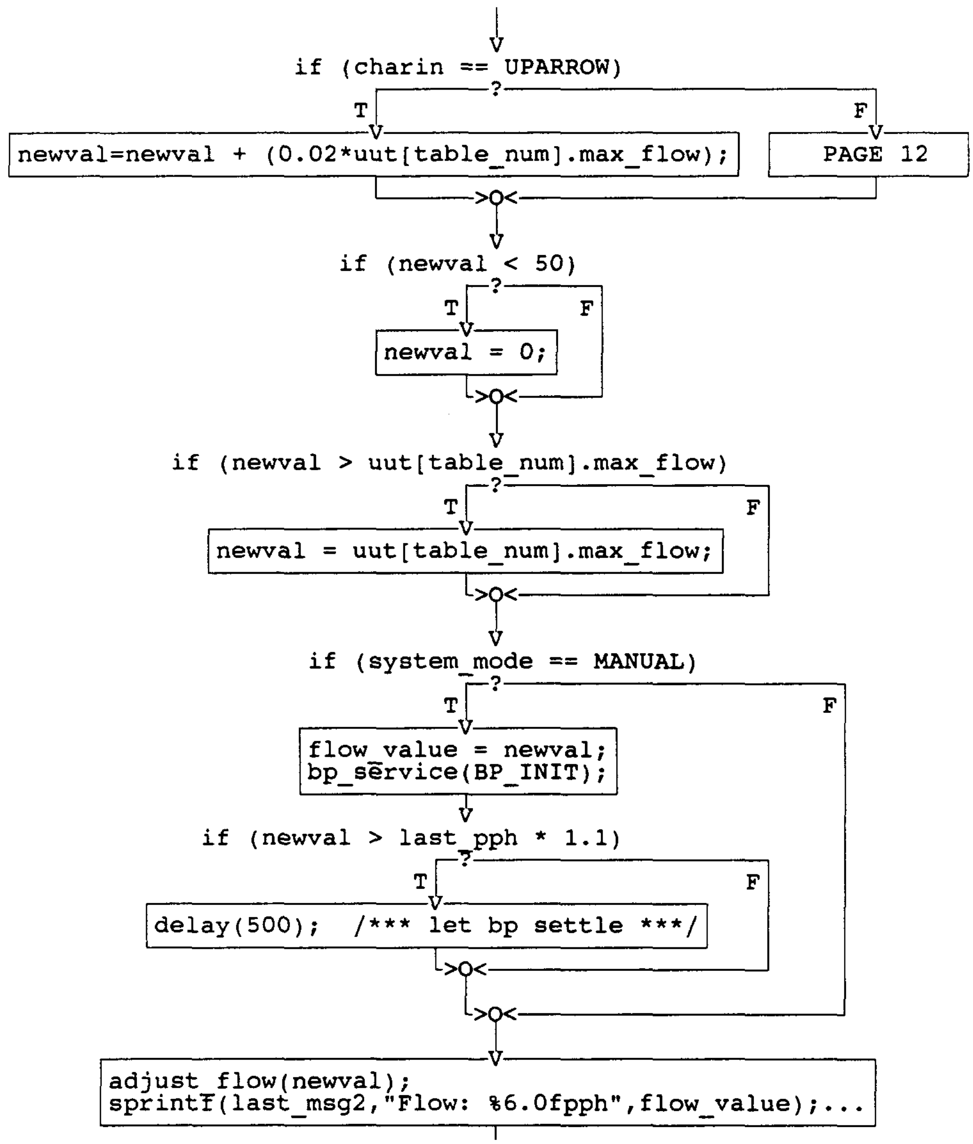

for (i=0;i<18;i++)
T
F
if (keyin2[i] == NULL)
T
F
if (i)
T
F
keyin2[i-1]=NULL;
break;
disp(2,keyin2);
break;

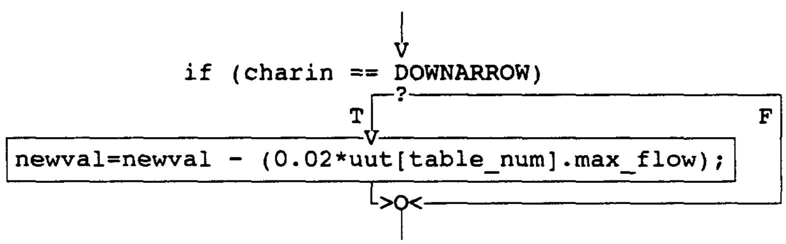
```
disp2_mode = FLOWSET;
reset_disp2();
sprintf(last_msg2,"Flow: %6.0fpph",flow_value);...
disp(2,last_msg2);
newval=flow_value;
<-break;
```

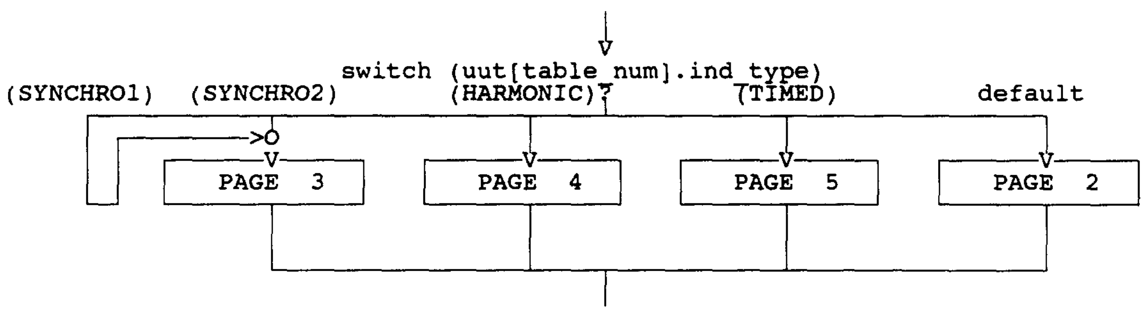
```
keyin2[0]=NULL;
disp(2,keyin2);
gpib_fault = 0;  /*** if fault message, clear f...
<-break;
```

```
                    |
                    V
       if (system_mode == MANUAL)
                   ?
                 T |                              F
                   V
 disp2_mode = PRESSURESET;
 reset_disp2();
 sprintf(last_msg2,"BkPress: %dpsi",bp_val);  /*...
 disp(2,last_msg2);
 newval=bp_val;
                   >O<
                    V
                <-break;
```

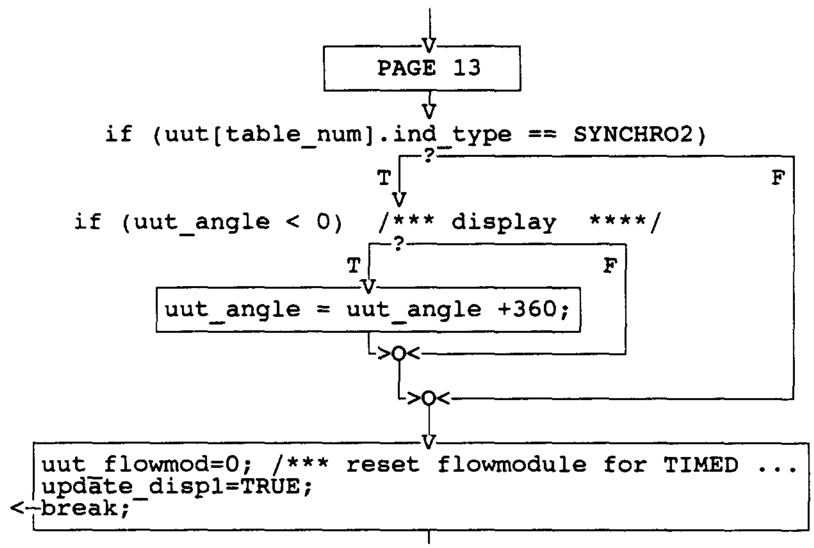

if (disp2_mode == FLOWSET)
T
F
PAGE 11
if (charin == UPARROW)
T
F
newval=newval + 1;
if (charin == DOWNARROW)
T
F
newval=newval - 1;
if (newval < 0)
T
F
newval = 0;
if (newval > 20)
T
F
newval = 20;
bp_val=(int)newval;
sprintf(last_msg2,"BkPress: %dpsi",bp_val); /*...
bp_service(BP_INIT); /*** set initial backpress...
disp(2,last_msg2);
reset_disp2();
break;


if (charin == UPARROW)
T
F
newval=newval + (0.02*uut[table_num].max_flow);
PAGE 12
if (newval < 50)
T
F
newval = 0;
if (newval > uut[table_num].max_flow)
T
F
newval = uut[table_num].max_flow;
if (system_mode == MANUAL)
T
F
flow_value = newval;
bp_service(BP_INIT);
if (newval > last_pph * 1.1)
T
F
delay(500); /*** let bp settle ***/
adjust_flow(newval);
sprintf(last_msg2,"Flow: %6.0fpph",flow_value);...

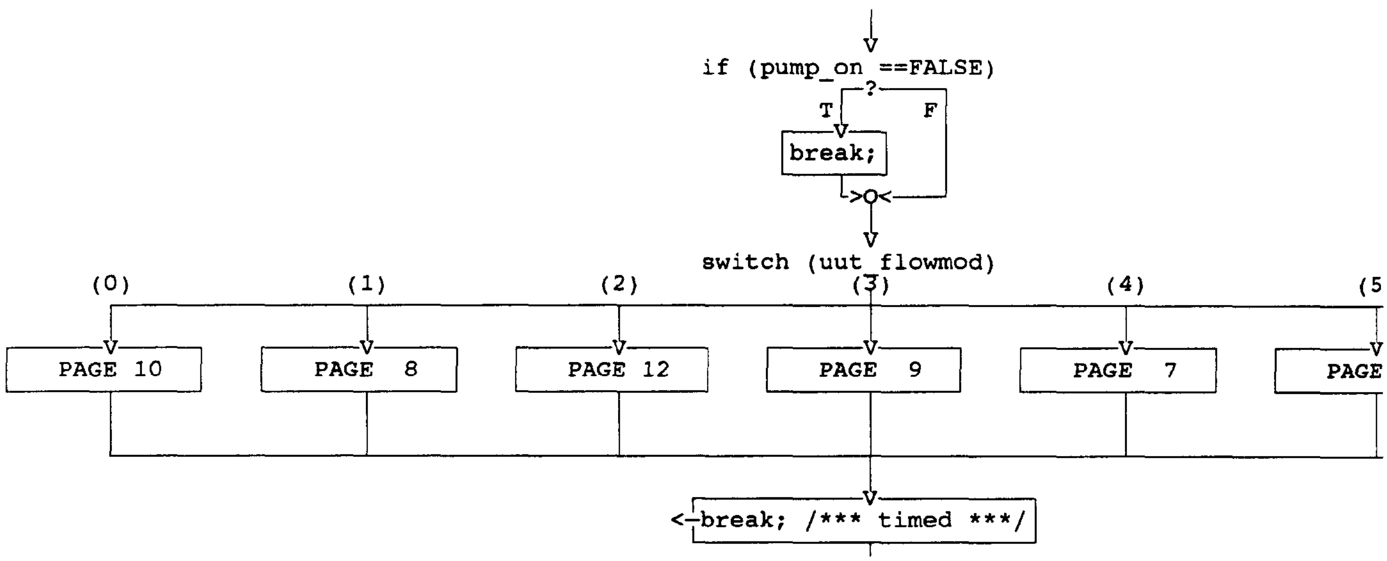

if (charin == DOWNARROW)
?
T
F
newval=newval - (0.02*uut[table_num].max_flow);


switch (uut[table_num].ind_type)
(SYNCHRO1)
(SYNCHRO2)
(HARMONIC)
(TIMED)
default
PAGE 3
PAGE 4
PAGE 5
PAGE 2

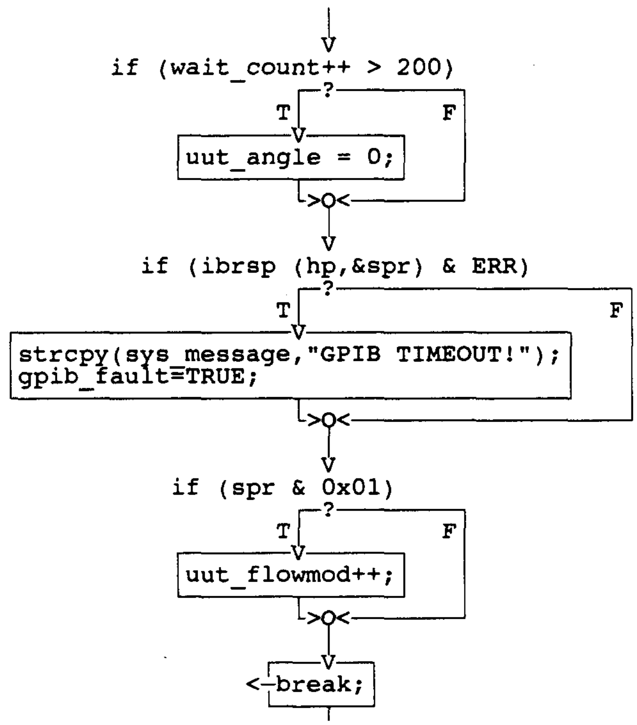
```
strcpy (sys_message,"Wrong UUT TYPE");
<-break;
```

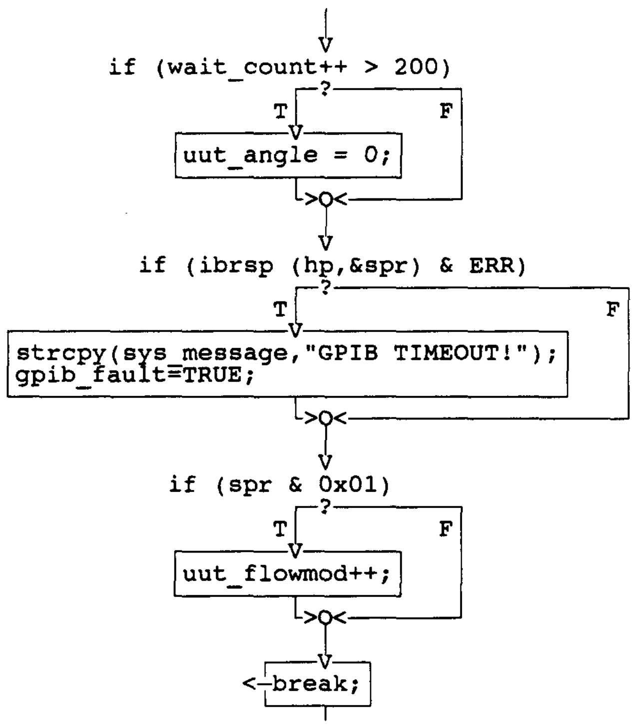

PAGE 13
if (uut[table_num].ind_type == SYNCHRO2)
?
T
F
if (uut_angle < 0) /*** display ****/
?
T
F
uut_angle = uut_angle +360;
uut_flowmod=0; /*** reset flowmodule for TIMED ...
update_displ=TRUE;
break;

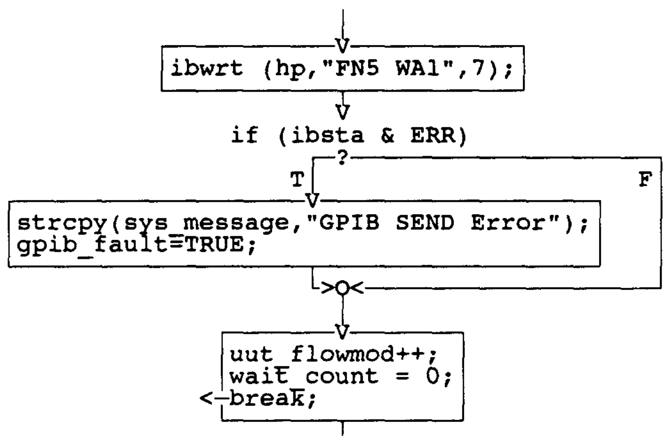
```
                                    |
                                    V
if (((temp = (thistime = biostime(0,0)) - lasttime)) < 10...
                                   ?
                                 T |                     F
                                   V
                     if (temp >= 0)
                                 ?
                               T |                 F
                                 V
 <-break; /*** wait .6 sec between reads ***/
                                 >O<
                                    >O<
                                      V
  lasttime = thistime;
  /*** process harmonic using 115vac input as ref...
  uut_angle = analog_in(HARMONICCHAN)*analog_in(H...
  + sys.harmonic_offset;
  uut_linearize(uut_angle);
  update_displ=TRUE;
  uut_flowmod=0; /*** reset flowmodule for TIMED ...
 <-break;
```

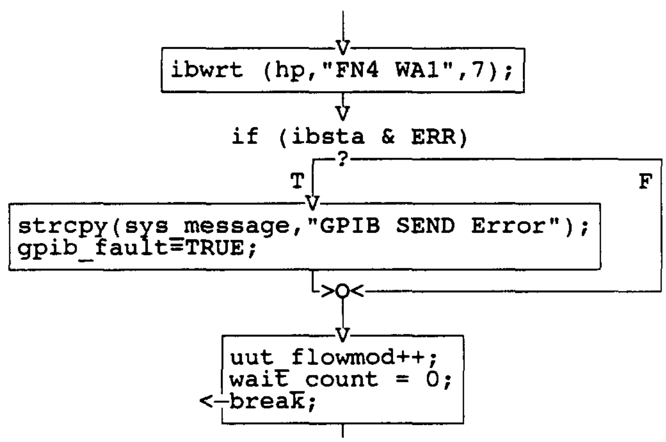

if (pump_on ==FALSE)
?
T
F
break;
switch (uut_flowmod)
(0)
(1)
(2)
(3)
(4)
(5
PAGE 10
PAGE 8
PAGE 12
PAGE 9
PAGE 7
PAGE
break; /*** timed ***/

```
uut_flowmod = 0;
<-break;
```


if (wait_count++ > 200)
T
F
uut_angle = 0;
if (ibrsp (hp,&spr) & ERR)
T
F
strcpy(sys_message,"GPIB TIMEOUT!");
gpib_fault=TRUE;
if (spr & 0x01)
T
F
uut_flowmod++;
break;

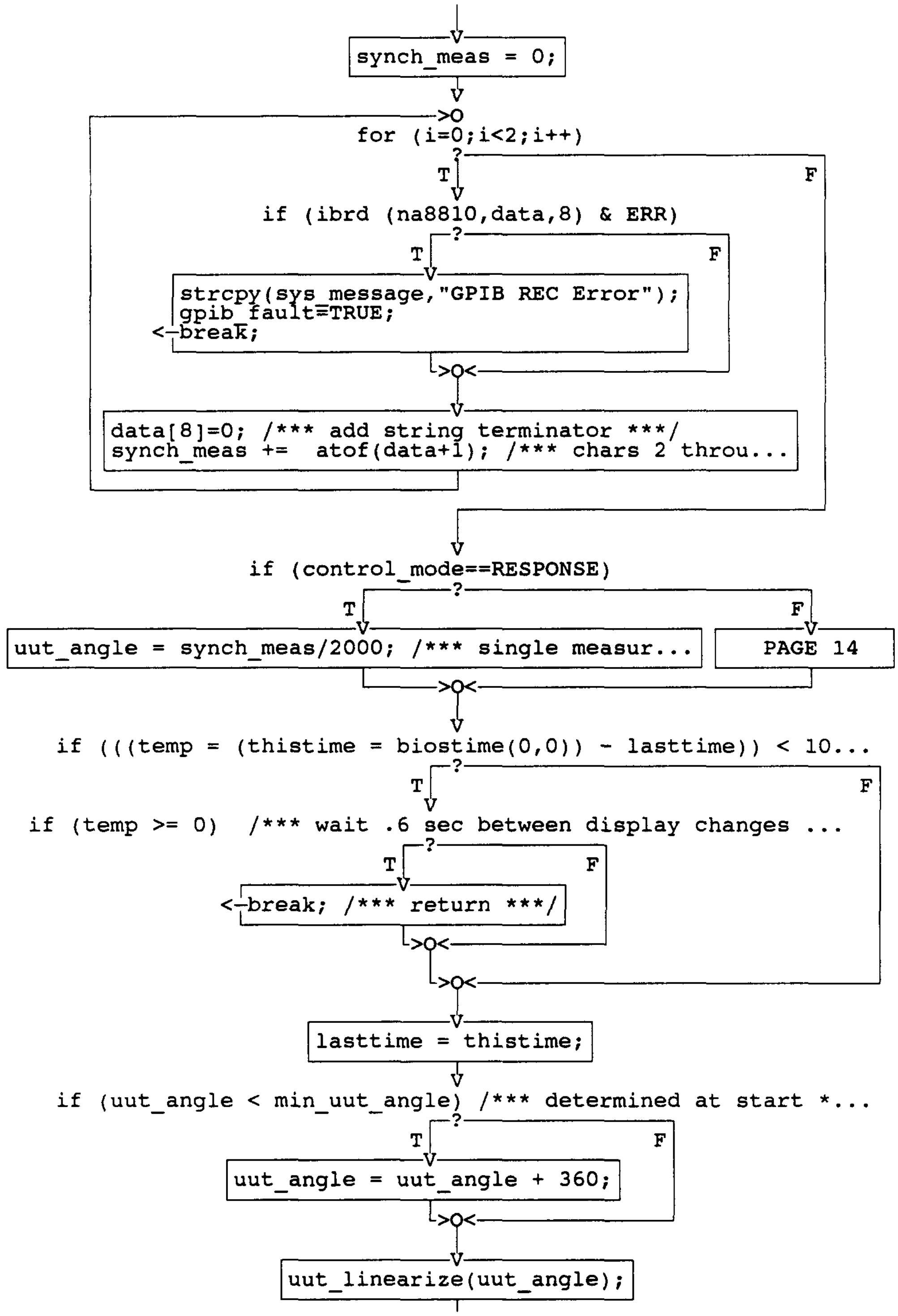

if (wait_count++ > 200)
?
T
F
uut_angle = 0;
if (ibrsp (hp,&spr) & ERR)
?
T
F
strcpy(sys_message,"GPIB TIMEOUT!");
gpib_fault=TRUE;
if (spr & 0x01)
?
T
F
uut_flowmod++;
break;


ibwrt (hp,"FN5 WA1",7);
if (ibsta & ERR)
?
T
F
strcpy(sys_message,"GPIB SEND Error");
gpib_fault=TRUE;
uut_flowmod++;
wait_count = 0;
break;

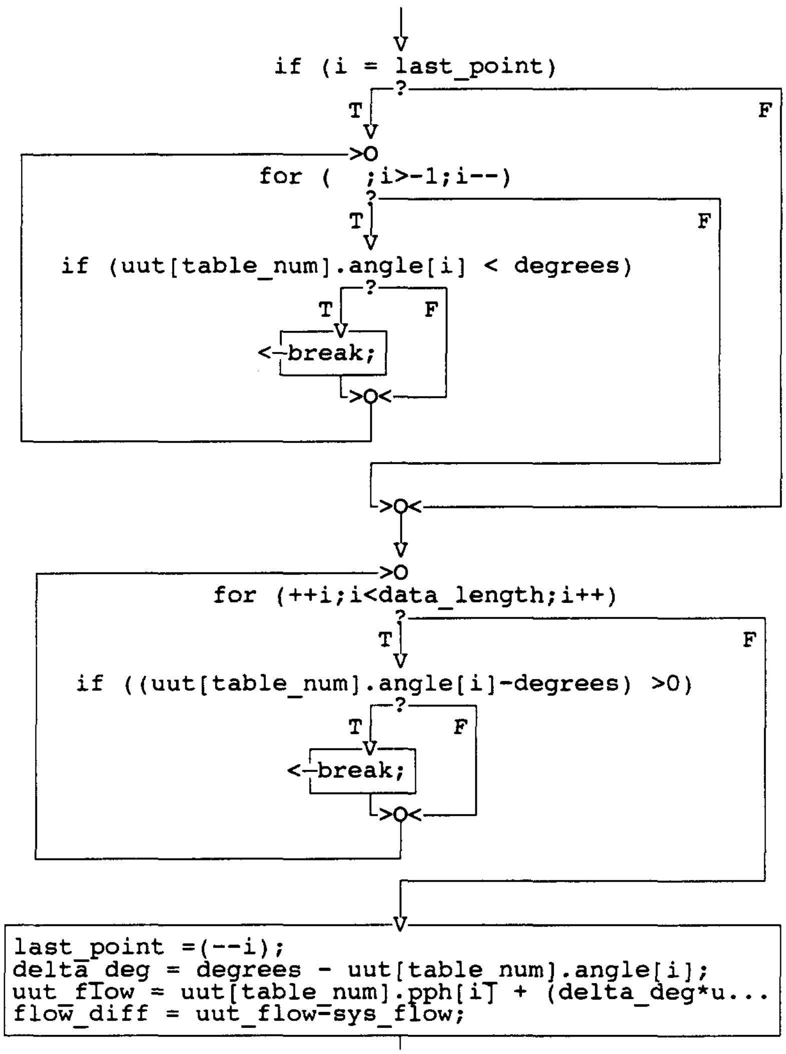

ibwrt (hp,"FN4 WA1",7);
if (ibsta & ERR)
?
T
F
strcpy(sys_message,"GPIB SEND Error");
gpib_fault=TRUE;
uut_flowmod++;
wait_count = 0;
break;

```
if (ibrd (hp,data,21) & ERR)
T                                        F
strcpy(sys_message,"GPIB REC Error");
gpib_fault=TRUE;

data[19]=0; /*** add string terminator at char ...
interval=atof(data+1); /*** convert character t...
/*** send period request ***/
ibwrt (hp,"FN4 WA1",7);

if (ibsta & ERR)
T                                        F
strcpy(sys_message,"GPIB SEND Error");
gpib_fault=TRUE;

wait_count = 0;
/*** compute angle ***/
uut_angle =((sys.drum_gain*interval/period) + s...
timed_avg = (timed_avg + uut_angle)-timed_avg/3;
uut_angle = timed_avg/3;
uut_linearize(uut_angle);
/*** restart at case 1 & flag update complete ***/
uut_flowmod=1;
update_displ=TRUE;
<-break;
```

```
if (ibrd (hp,data,21) & ERR)
T                                              F
strcpy(sys_message,"GPIB REC Error");
gpib_fault=TRUE;

data[19]=0; /*** add string terminator at char ...
period=atof(data+1); /*** convert from characte...
uut_flowmod++;
break;
```

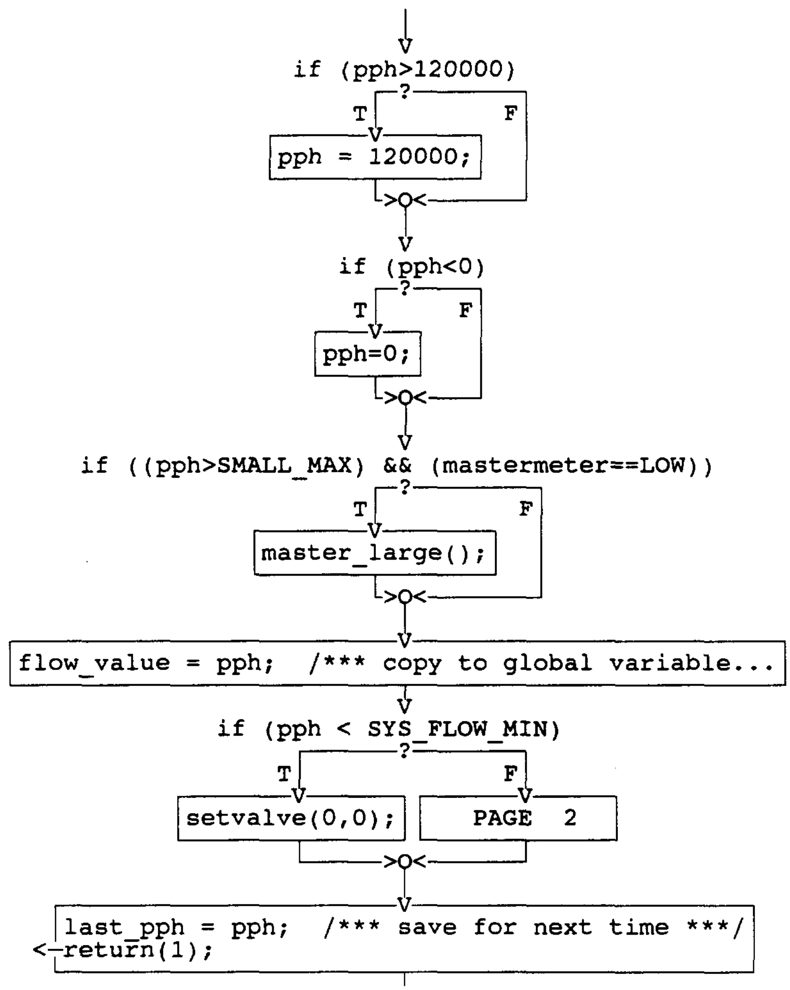

synch_meas = 0;
for (i=0;i<2;i++)
T
F
if (ibrd (na8810,data,8) & ERR)
T
F
strcpy(sys_message,"GPIB REC Error");
gpib_fault=TRUE;
break;
data[8]=0; /*** add string terminator ***/
synch_meas += atof(data+1); /*** chars 2 throu...
if (control_mode==RESPONSE)
T
F
uut_angle = synch_meas/2000; /*** single measur...
PAGE 14
if (((temp = (thistime = biostime(0,0)) - lasttime)) < 10...
T
F
if (temp >= 0) /*** wait .6 sec between display changes ...
T
F
break; /*** return ***/
lasttime = thistime;
if (uut_angle < min_uut_angle) /*** determined at start *...
T
F
uut_angle = uut_angle + 360;
uut_linearize(uut_angle);

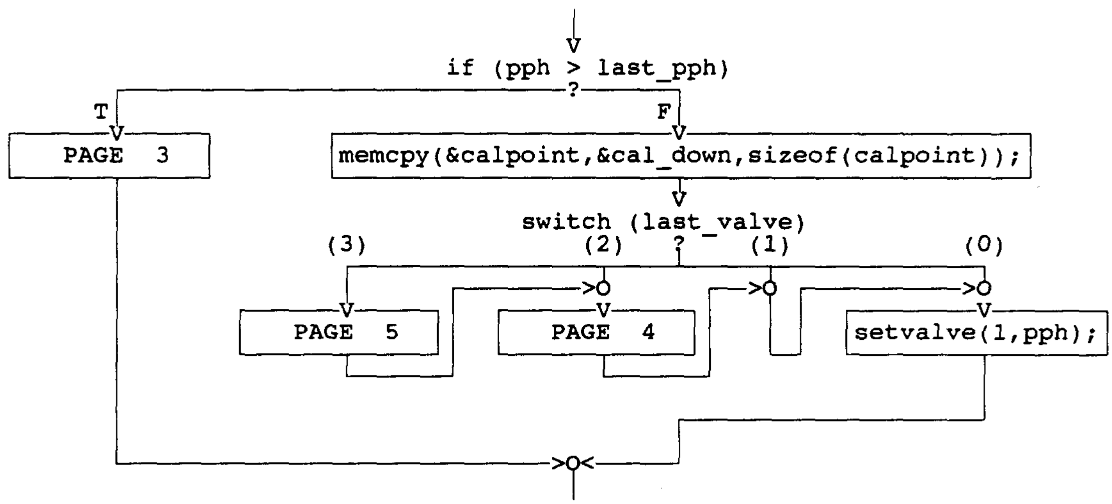
```
uut_angle = ((synch_avg=synch_avg-(synch_avg/20...
/*** remember 20*2000 ***/
```

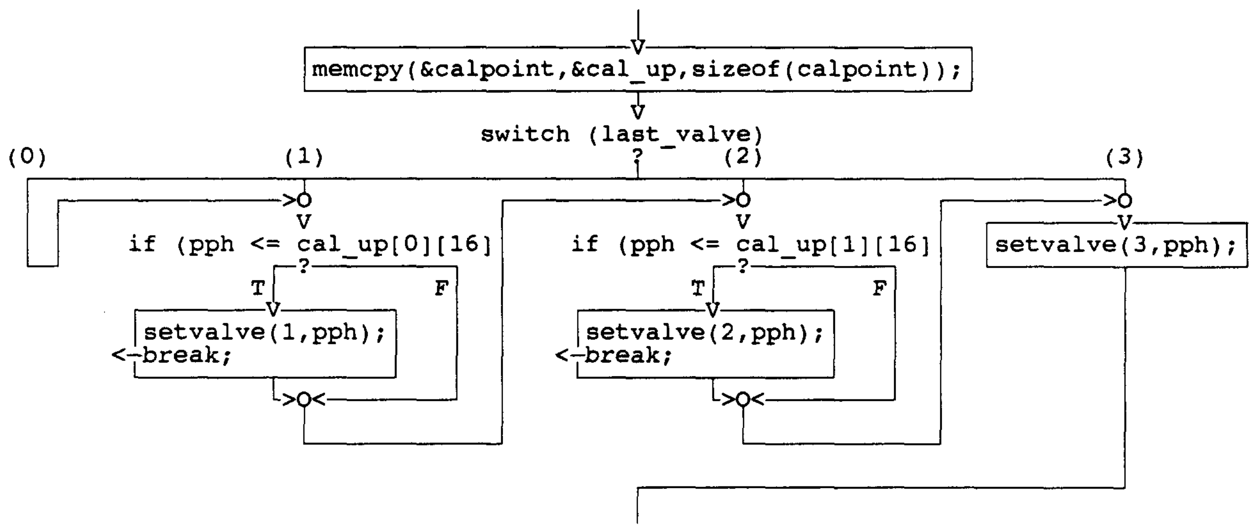

if (i = last_point)
T
F
for (;i>-1;i--)
T
F
if (uut[table_num].angle[i] < degrees)
T
F
break;
for (++i;i<data_length;i++)
T
F
if ((uut[table_num].angle[i]-degrees) >0)
T
F
break;
last_point =(--i);
delta_deg = degrees - uut[table_num].angle[i];
uut_flow = uut[table_num].pph[i] + (delta_deg*u...
flow_diff = uut_flow-sys_flow;

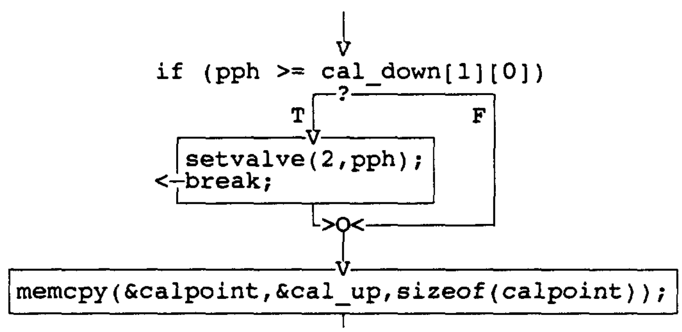
```
memcpy(&uut_temp,&uut[table_num],sizeof(struct ...
memcpy(&sys_temp,&sys,sizeof(struct sys_table));
memcpy(&temp_calup,&cal_up,sizeof(cal_up));
memcpy(&temp_caldown,&cal_down,sizeof(cal_down));
memcpy(&temp_calback,&cal_back,sizeof(cal_back));
#ifdef DEBUG

for (i=0;i<17;i++)
?
T                                              F

printf("%d  %f\n",i,1000000*cal_back[i]);

#endif
```

```
memcpy(&uut[table_num],&uut_temp,sizeof(struct ...
memcpy(&sys,&sys_temp,sizeof(struct sys_table));
memcpy(&cal_up,&temp_calup,sizeof(cal_up));
memcpy(&cal_down,&temp_caldown,sizeof(cal_down));

memcpy(&cal_back,&temp_calback,sizeof(cal_back));
#ifdef DEBUG

for (i=0;i<17;i++)
  T
    printf("%d  %f\n",i,100000*cal_back[i]);

    #endif
  F
```


if (pph>120000)
?
T
F
pph = 120000;
if (pph<0)
?
T
F
pph=0;
if ((pph>SMALL_MAX) && (mastermeter==LOW))
?
T
F
master_large();
flow_value = pph; /*** copy to global variable...
if (pph < SYS_FLOW_MIN)
?
T
F
setvalve(0,0);
PAGE 2
last_pph = pph; /*** save for next time ***/
return(1);


if (pph > last_pph)
T
F
PAGE 3
memcpy(&calpoint,&cal_down,sizeof(calpoint));
switch (last_valve)
(3)
(2)
(1)
(0)
PAGE 5
PAGE 4
setvalve(1,pph);

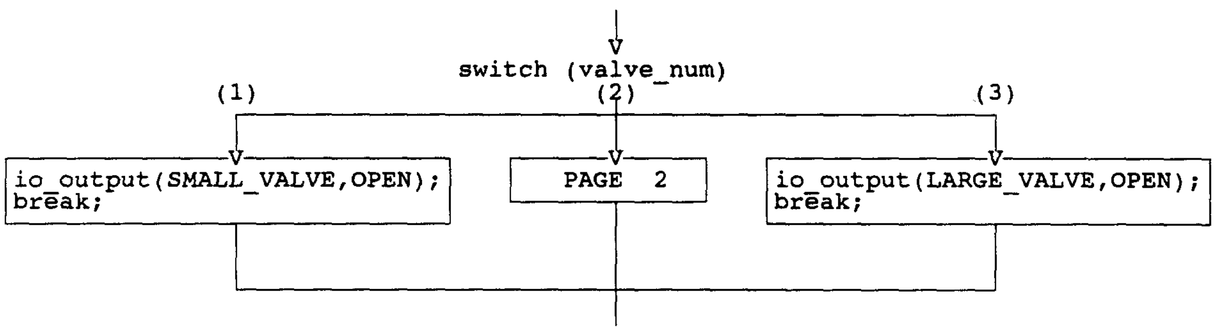

memcpy(&calpoint,&cal_up,sizeof(calpoint));
switch (last_valve)
(0)
(1)
(2)
(3)
if (pph <= cal_up[0][16]
T
F
setvalve(1,pph);
break;
if (pph <= cal_up[1][16]
T
F
setvalve(2,pph);
break;
setvalve(3,pph);


if (pph >= cal_down[1][0])
?
T
F
setvalve(2,pph);
break;
memcpy(&calpoint,&cal_up,sizeof(calpoint));

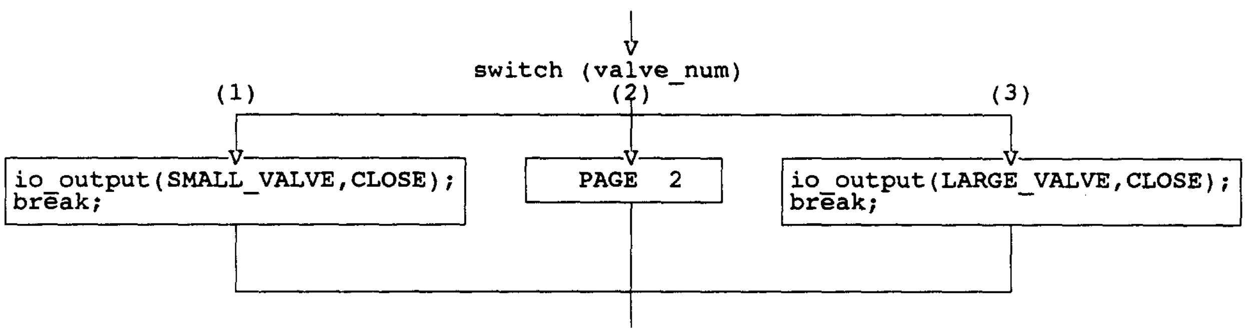
```
                 |
                 V
if (pph >= cal_down[1][16]-1000)
                ?
              T   F
              V
     setvalve(3,pph);
  <--break;

              >O<
               V
memcpy(&calpoint,&cal_up,sizeof(calpoint));
```

```
PAGE 2

if (valve_num--)
T
current = 0;  /*** default if table out of rang...
F

for (dp=0;dp<16;dp++)
T
F
if (flow_value < calpoint[valve_num][dp+1])
T
F
current = dp + ((flow_value -calpoint[valve_num...
/(calpoint[valve_num][dp+1]-calpoint[valve_num]...
<-break;

ma_out(valve_io[valve_num],current);
vopen (valve_num +1);

if (press_turnoff)
T
F
delay(1000);
ma_out(BKPRESS_OUT,back_ma); /*** reset previou...

if (++valve_num != last_valve)
T
F
delay(1500);  /**msec for turnon surge to pass ...
disp(2,"READY");

if (!bypass_on)
T
F
turn_bypass(OFF);

last_valve = valve_num;
```

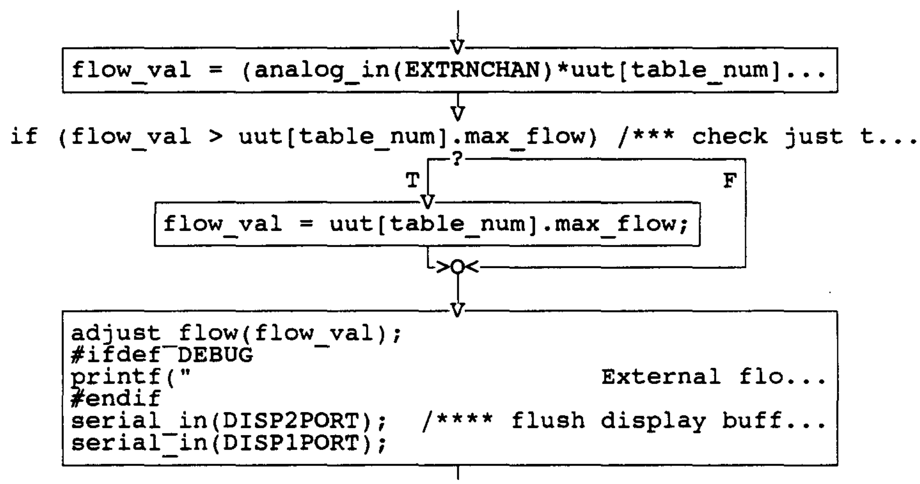
```
if ((flow > SMALL_MAX) && (mastermeter == LOW))
    ?
  T                                                     F
    disp(2,"Flow Too HIGH!!");
    delay (2000); /*** lock up every thing for 2 se...
    flow = SMALL_MAX;
  >O<

if (valve_num == 0)
    ?
  T                                                     F
    vclose(1);
    vclose(2);
    vclose(3);
    ma_out(valve_io[0],0); /*** set all valves to m...
    ma_out(valve_io[1],0);
    ma_out(valve_io[2],0);
  >O<

if (valve_num != last_valve)
    ?
  T                                                     F
    io_output(BYPASS_VALVE,ON); /*** bypass turnon ...
    disp(2,"CHANGING VALVES");
    vclose(1); /*** if different flow valve shut al...
    vclose(2);
    vclose(3);
    ma_out(valve_io[0],0); /*** set all valves to m...
    ma_out(valve_io[1],0);
    ma_out(valve_io[2],0);

    if (valve_num >= 2)
        ?
      T                                                 F
        ma_out(BKPRESS_OUT,0); /*** turnoff backpressur...
        delay (1000);
        press_turnoff = TRUE;
      >O<
  >O<
```

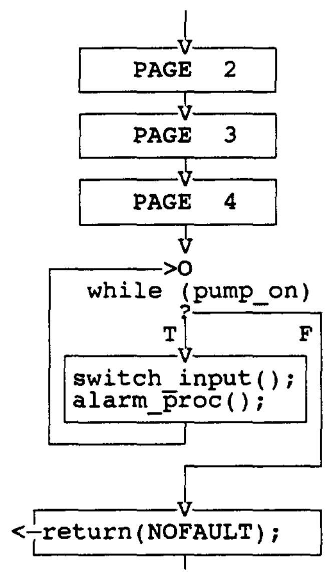

switch (valve_num)
(1)
(2)
(3)
io_output(SMALL_VALVE,OPEN);
break;
PAGE 2
io_output(LARGE_VALVE,OPEN);
break;

```
io_output(MEDIUM_VALVE,OPEN);
break;
```

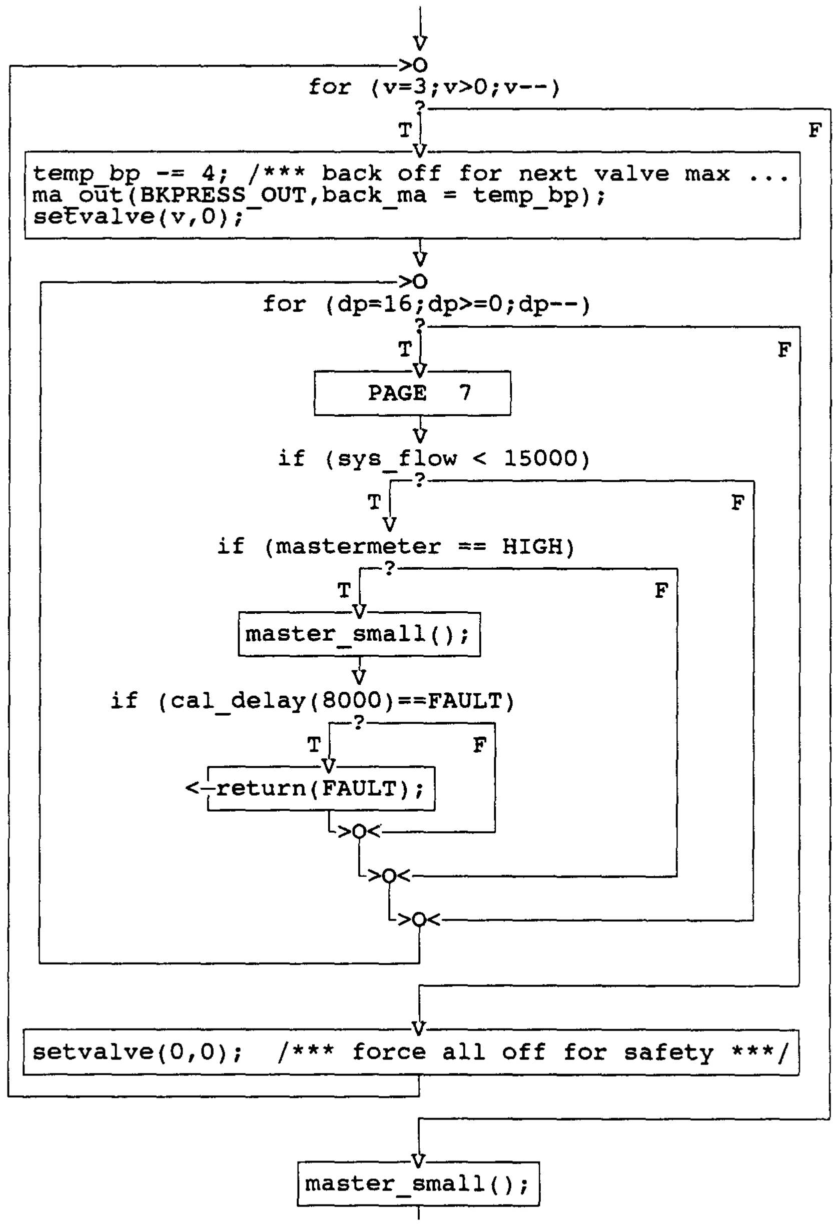

switch (valve_num)
(1)
(2)
(3)
io_output(SMALL_VALVE,CLOSE);
break;
PAGE 2
io_output(LARGE_VALVE,CLOSE);
break;

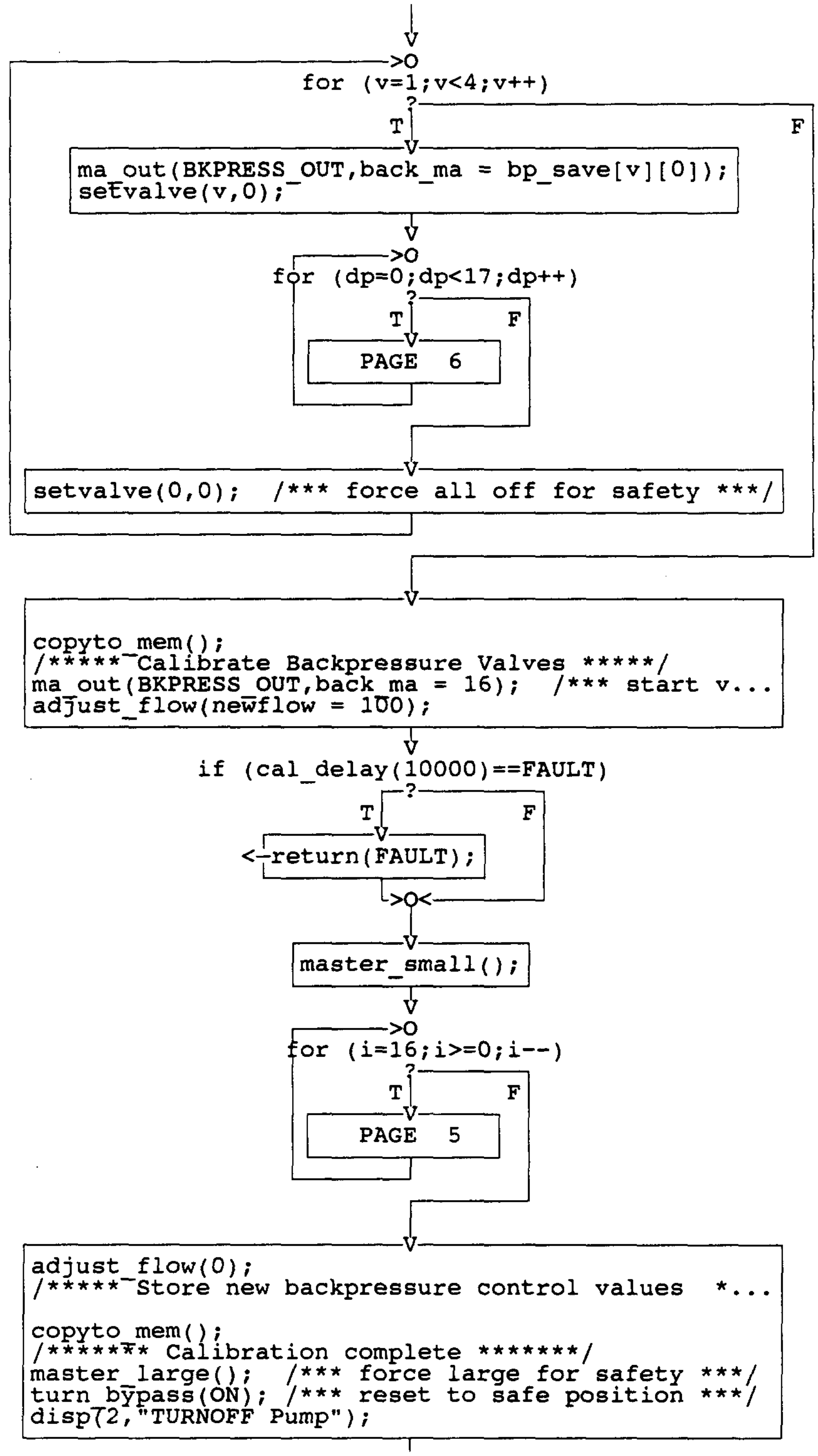
```
io_output(MEDIUM_VALVE,CLOSE);
<-break;
```

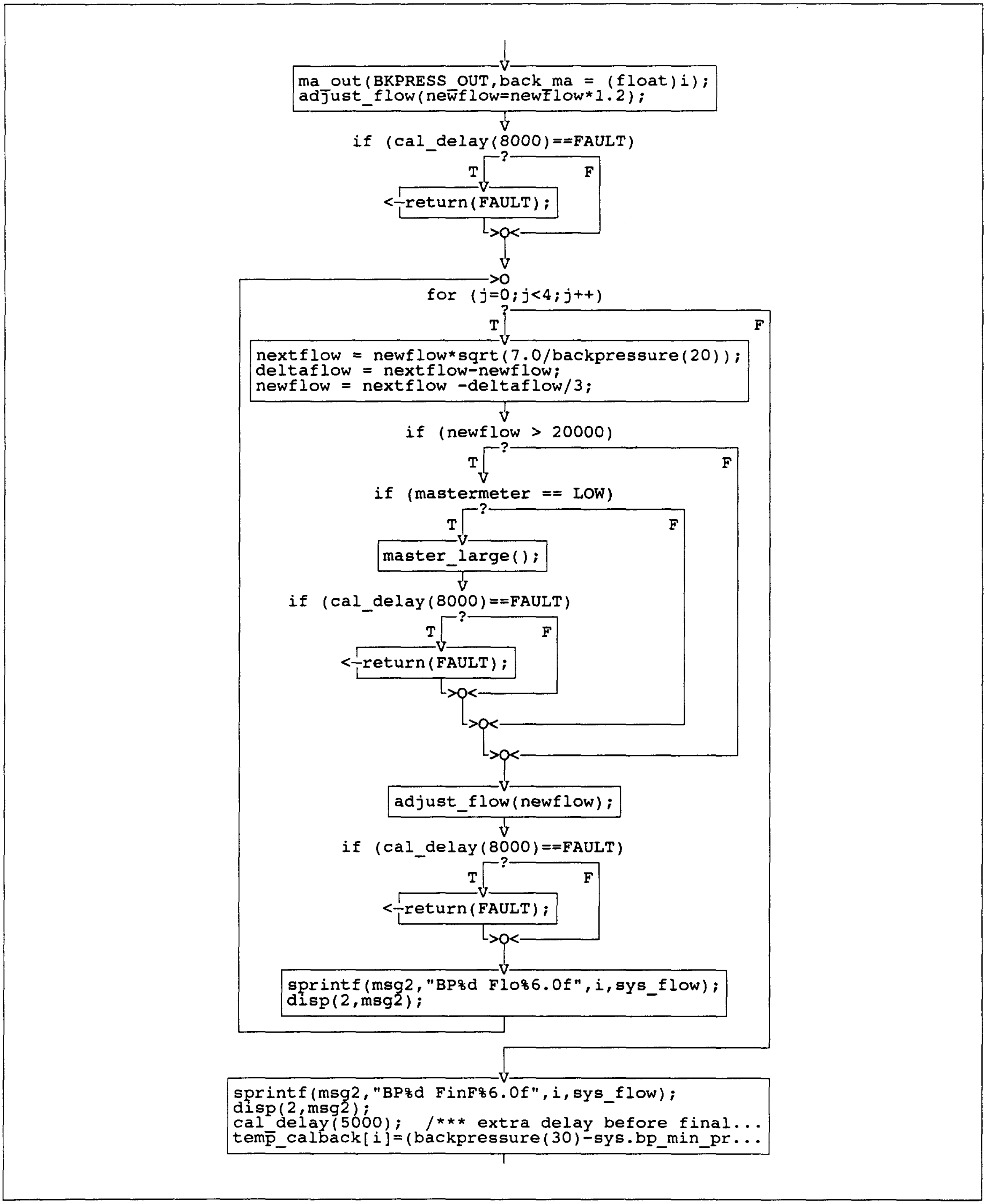

flow_val = (analog_in(EXTRNCHAN)*uut[table_num]...
if (flow_val > uut[table_num].max_flow) /*** check just t...
?
T
F
flow_val = uut[table_num].max_flow;
adjust_flow(flow_val);
#ifdef DEBUG
printf("   External flo...
#endif
serial_in(DISP2PORT); /**** flush display buff...
serial_in(DISP1PORT);

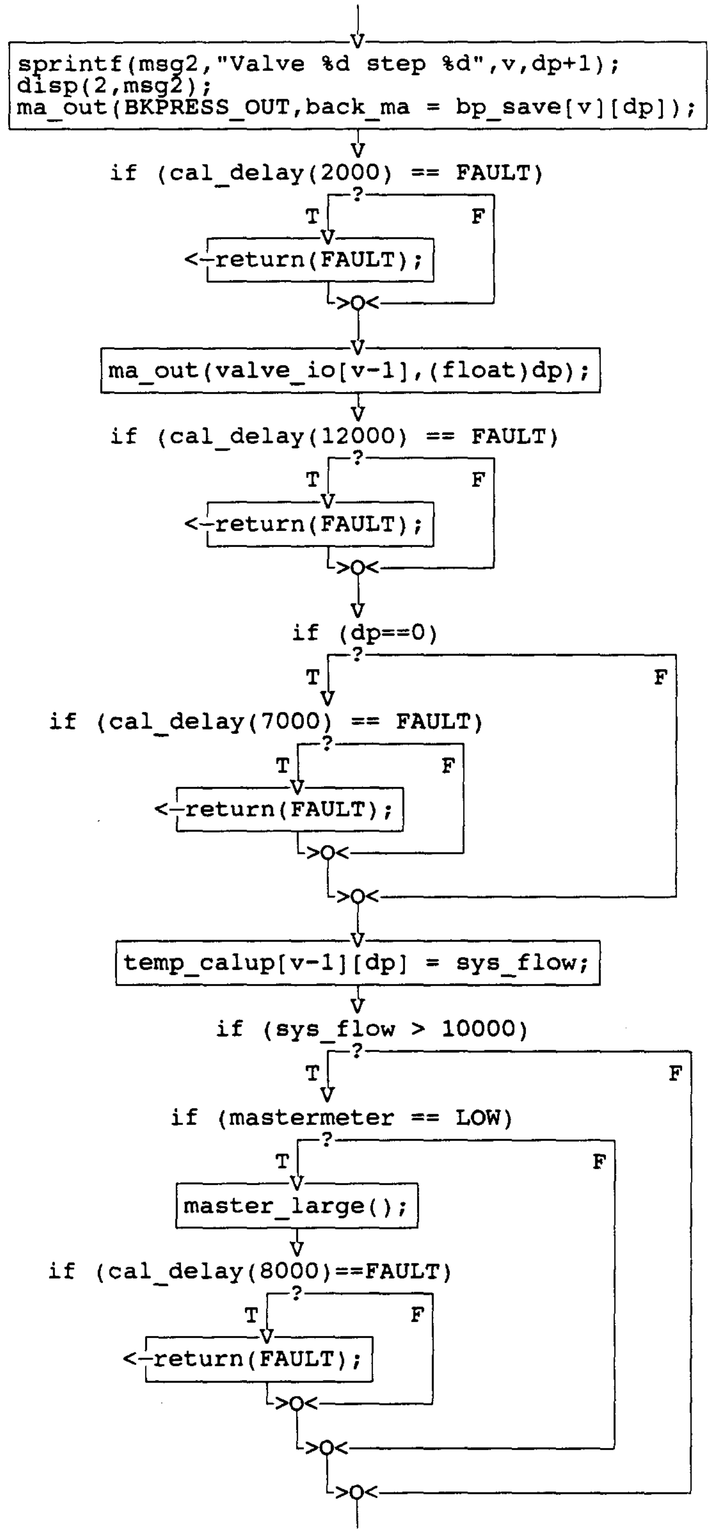

PAGE 2
PAGE 3
PAGE 4
while (pump_on)
T
F
switch_input();
alarm_proc();
return(NOFAULT);

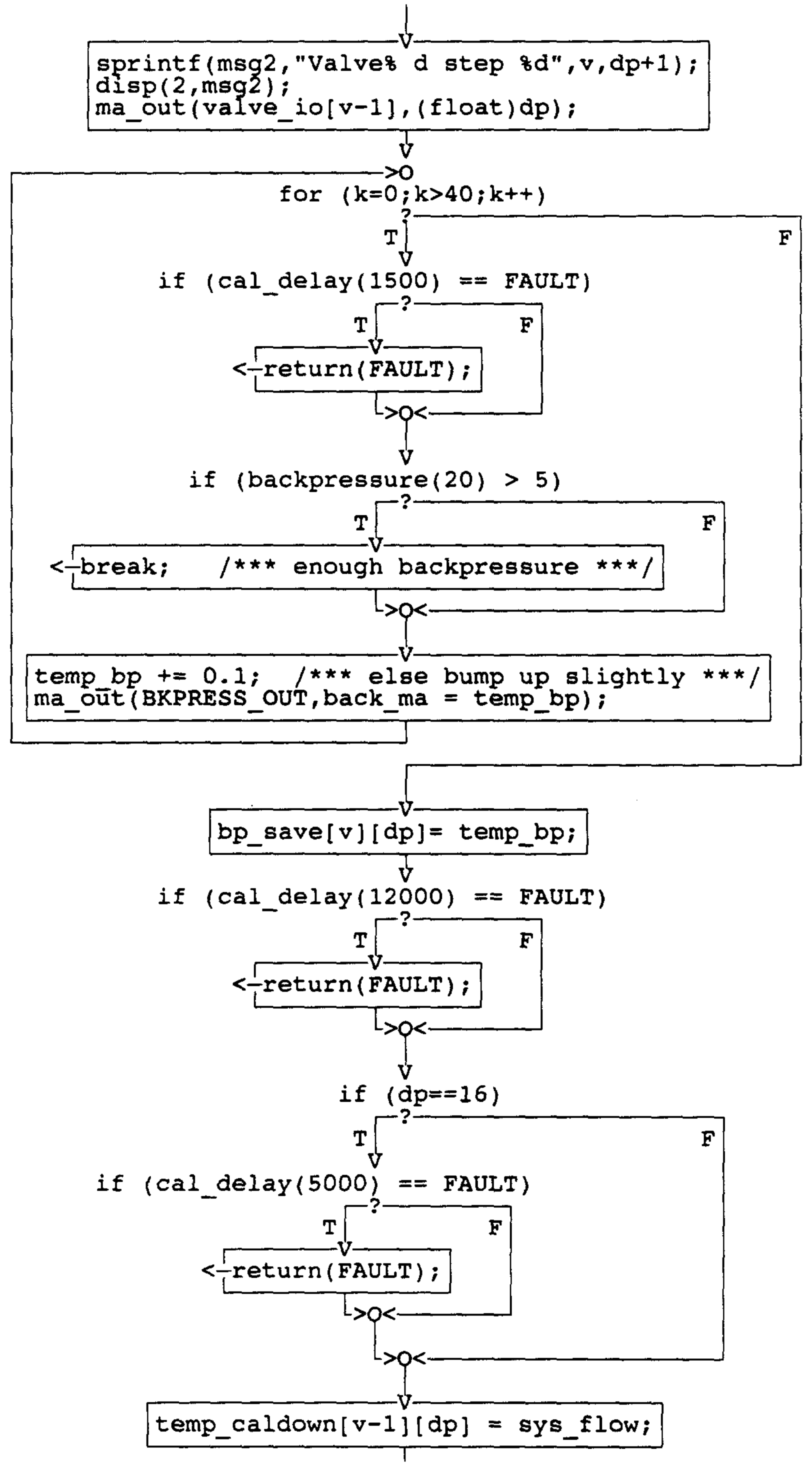
```
setvalve(0,0);   /*** initialize all valves off...
ma_out(BKPRESS_OUT,back_ma = 0);
disp(2,"TURNON Pump");

while (pump_on ==FALSE)
  T
    switch_input();
    alarm_proc();

    if (prog_mode ==OFF)
      T
        <-return(FAULT);
      F
  F

turn_bypass(OFF);  /*** insure no bypass ***/
/*** Calibrate from max to min flow ***/
master_large();
temp_bp = 4;
```

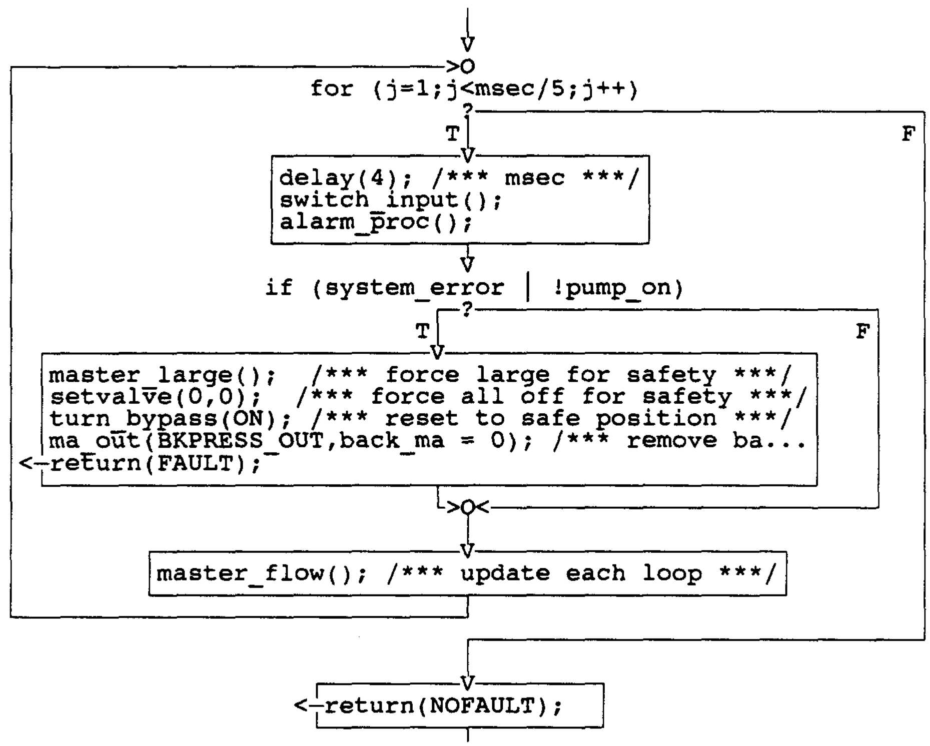

for (v=3;v>0;v--)
T
F
temp_bp -= 4; /*** back off for next valve max ...
ma_out(BKPRESS_OUT,back_ma = temp_bp);
setvalve(v,0);
for (dp=16;dp>=0;dp--)
T
F
PAGE 7
if (sys_flow < 15000)
T
F
if (mastermeter == HIGH)
T
F
master_small();
if (cal_delay(8000)==FAULT)
T
F
return(FAULT);
setvalve(0,0); /*** force all off for safety ***/
master_small();


for (v=1;v<4;v++)
T
F
ma_out(BKPRESS_OUT,back_ma = bp_save[v][0]);
setvalve(v,0);
for (dp=0;dp<17;dp++)
T
F
PAGE 6
setvalve(0,0); /*** force all off for safety ***/
copyto_mem();
/***** Calibrate Backpressure Valves *****/
ma_out(BKPRESS_OUT,back_ma = 16); /*** start v...
adjust_flow(newflow = 100);
if (cal_delay(10000)==FAULT)
T
F
return(FAULT);
master_small();
for (i=16;i>=0;i--)
T
F
PAGE 5
adjust_flow(0);
/***** Store new backpressure control values *...
copyto_mem();
/******** Calibration complete *******/
master_large(); /*** force large for safety ***/
turn_bypass(ON); /*** reset to safe position ***/
disp(2,"TURNOFF Pump");

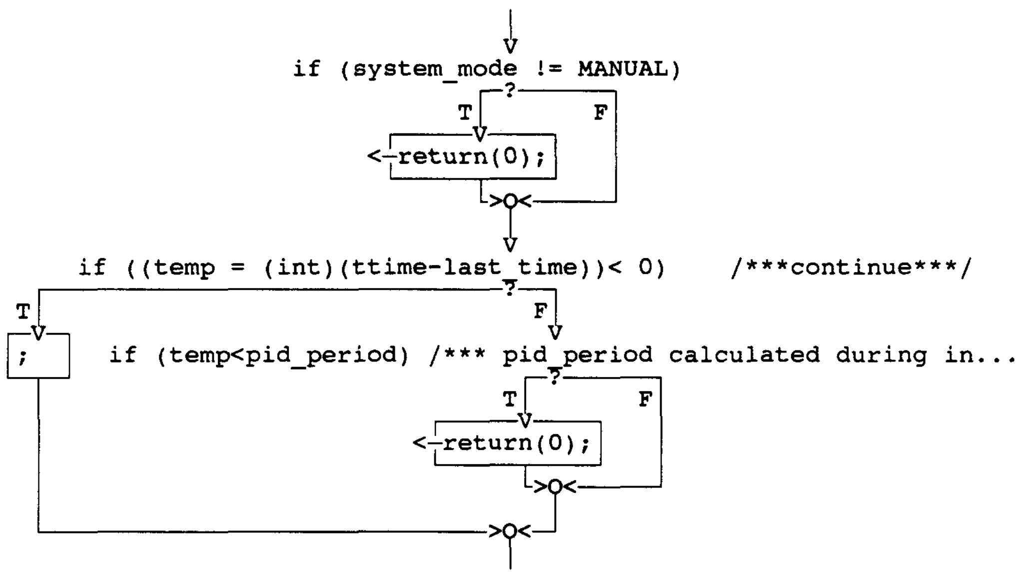

ma_out(BKPRESS_OUT,back_ma = (float)i);
adjust_flow(newflow=newflow*1.2);
if (cal_delay(8000)==FAULT)
T
F
return(FAULT);
for (j=0;j<4;j++)
T
F
nextflow = newflow*sqrt(7.0/backpressure(20));
deltaflow = nextflow-newflow;
newflow = nextflow -deltaflow/3;
if (newflow > 20000)
T
F
if (mastermeter == LOW)
T
F
master_large();
if (cal_delay(8000)==FAULT)
T
F
return(FAULT);
adjust_flow(newflow);
if (cal_delay(8000)==FAULT)
T
F
return(FAULT);
sprintf(msg2,"BP%d Flo%6.0f",i,sys_flow);
disp(2,msg2);
sprintf(msg2,"BP%d FinF%6.0f",i,sys_flow);
disp(2,msg2);
cal_delay(5000); /*** extra delay before final...
temp_calback[i]=(backpressure(30)-sys.bp_min_pr...

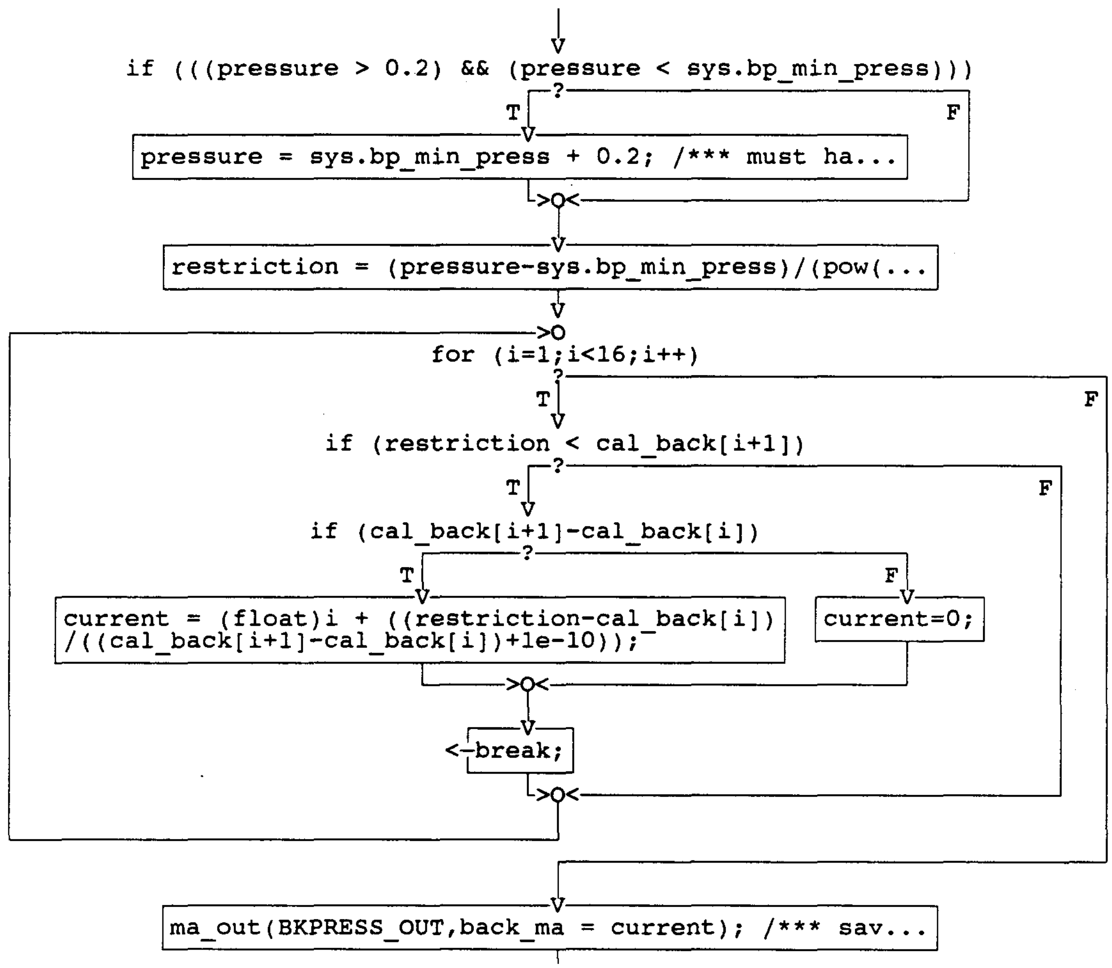

sprintf(msg2,"Valve %d step %d",v,dp+1);
disp(2,msg2);
ma_out(BKPRESS_OUT,back_ma = bp_save[v][dp]);
if (cal_delay(2000) == FAULT)
?
T
F
<-return(FAULT);
ma_out(valve_io[v-1],(float)dp);
if (cal_delay(12000) == FAULT)
?
T
F
<-return(FAULT);
if (dp==0)
?
T
F
if (cal_delay(7000) == FAULT)
?
T
F
<-return(FAULT);
temp_calup[v-1][dp] = sys_flow;
if (sys_flow > 10000)
?
T
F
if (mastermeter == LOW)
?
T
F
master_large();
if (cal_delay(8000)==FAULT)
?
T
F
<-return(FAULT);

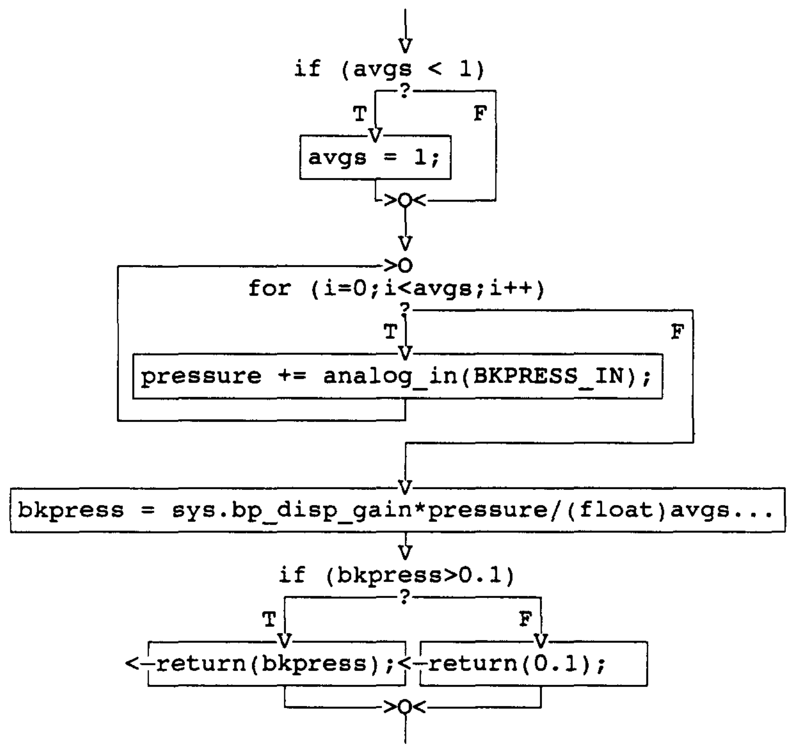

sprintf(msg2,"Valve% d step %d",v,dp+1);
disp(2,msg2);
ma_out(valve_io[v-1],(float)dp);
for (k=0;k>40;k++)
T
F
if (cal_delay(1500) == FAULT)
T
F
return(FAULT);
if (backpressure(20) > 5)
T
F
break; /*** enough backpressure ***/
temp_bp += 0.1; /*** else bump up slightly ***/
ma_out(BKPRESS_OUT,back_ma = temp_bp);
bp_save[v][dp]= temp_bp;
if (cal_delay(12000) == FAULT)
T
F
return(FAULT);
if (dp==16)
T
F
if (cal_delay(5000) == FAULT)
T
F
return(FAULT);
temp_caldown[v-1][dp] = sys_flow;

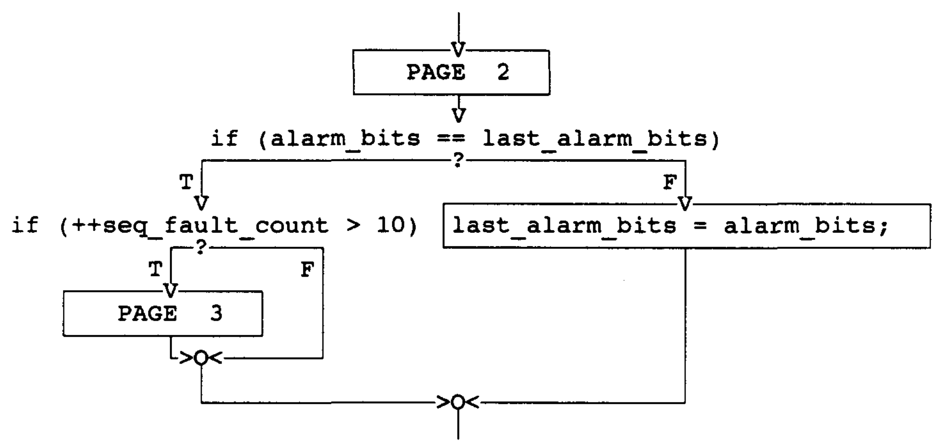

for (j=1;j<msec/5;j++)
T
F
delay(4); /*** msec ***/
switch_input();
alarm_proc();
if (system_error | !pump_on)
T
F
master_large(); /*** force large for safety ***/
setvalve(0,0); /*** force all off for safety ***/
turn_bypass(ON); /*** reset to safe position ***/
ma_out(BKPRESS_OUT,back_ma = 0); /*** remove ba...
return(FAULT);
master_flow(); /*** update each loop ***/
return(NOFAULT);

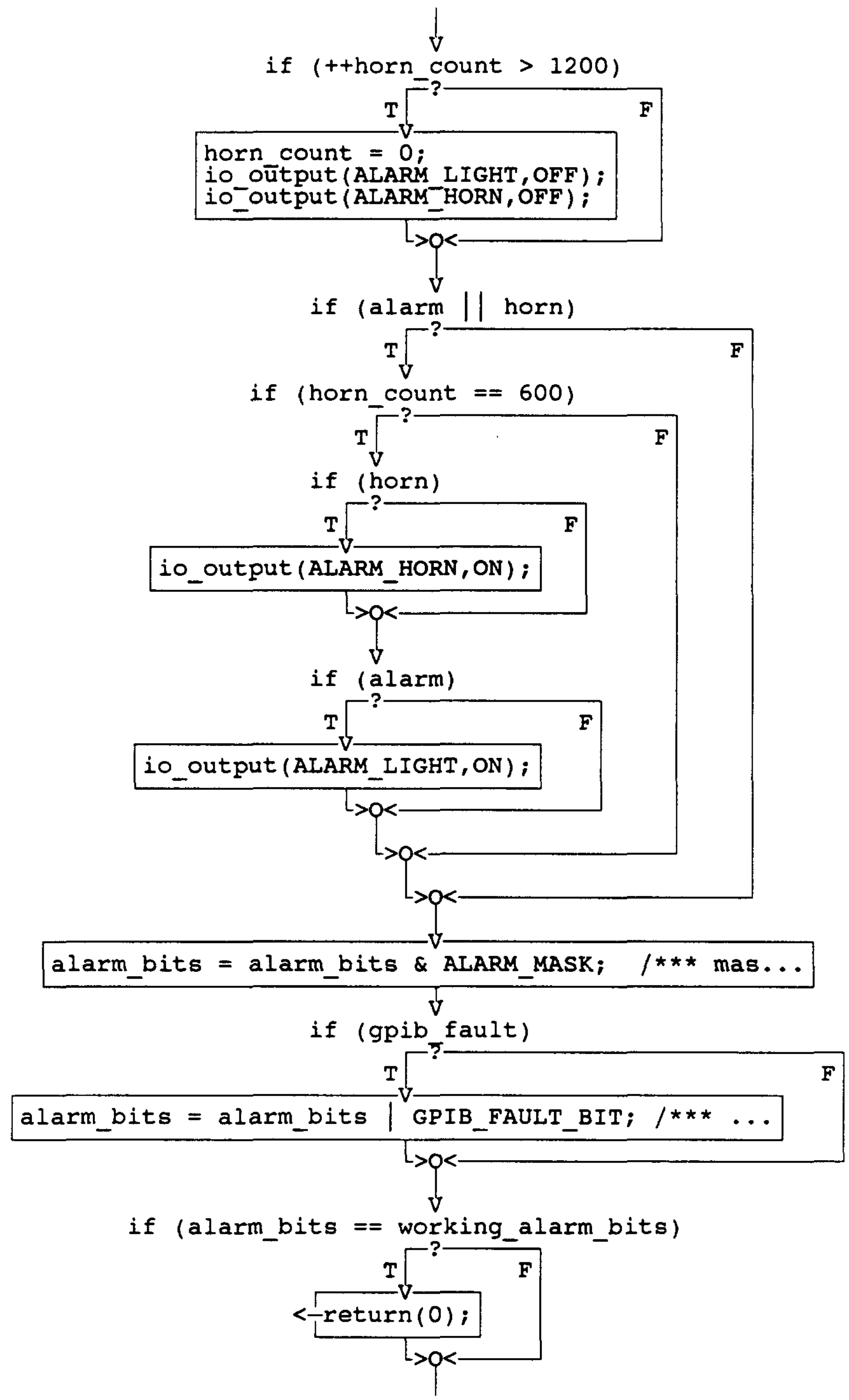
```
if ((flow_value < 40) || !pump_on || bypass_on)
    T
    <-return(0);
    F

if (init_cmd == BP_INIT)
    T
    bp_integral = bp_val/sys.bp_I;
    bp_set(bp_val);
    last_error = 0;
    last_time = ttime; /*** force update with new v...
    F
    PAGE  2

update = sys.bp_P*(error = bp_val - backpressur...
+ sys.bp_I*(bp_integral += sys.bp_Time*error)
+ sys.bp_D*(error-last_error);
#ifdef DEBUG
printf("err=%f  int=%f  dif=%f >%f  ",error,bp_...
#endif
last_error = error;
last_time = ttime;

if (init_cmd !=BP_INIT)
    T
    bp_set(update); /*** output new ma for adjusted...
    F
```

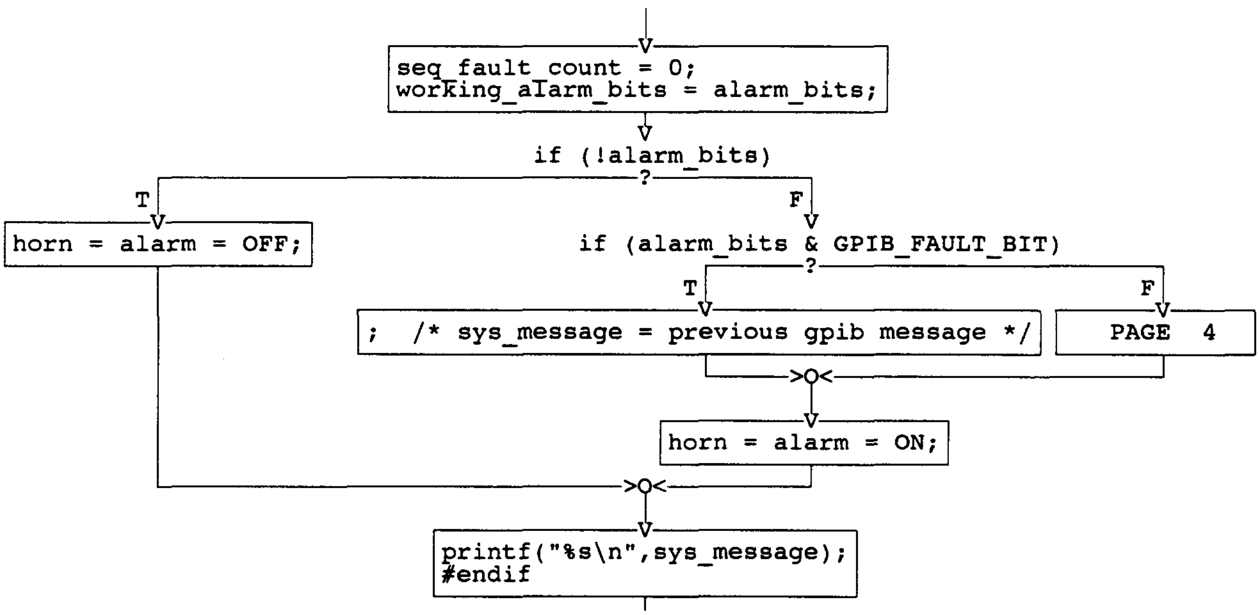

if (system_mode != MANUAL)
T
F
return(0);
if ((temp = (int)(ttime-last_time))< 0)
/***continue***/
T
F
;
if (temp<pid_period) /*** pid_period calculated during in...
T
F
return(0);

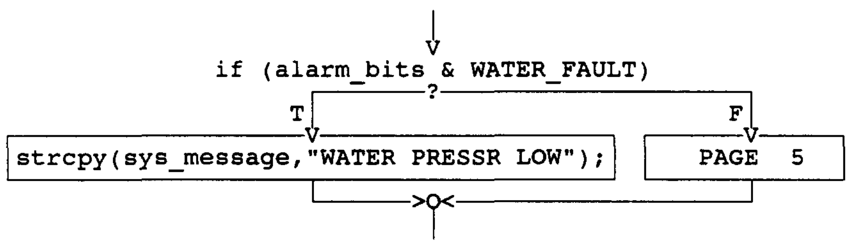

if (((pressure > 0.2) && (pressure < sys.bp_min_press)))
?
T
F
pressure = sys.bp_min_press + 0.2; /*** must ha...
restriction = (pressure-sys.bp_min_press)/(pow(...
for (i=1;i<16;i++)
?
T
F
if (restriction < cal_back[i+1])
?
T
F
if (cal_back[i+1]-cal_back[i])
?
T
F
current = (float)i + ((restriction-cal_back[i])
/((cal_back[i+1]-cal_back[i])+1e-10));
current=0;
break;
ma_out(BKPRESS_OUT,back_ma = current); /*** sav...

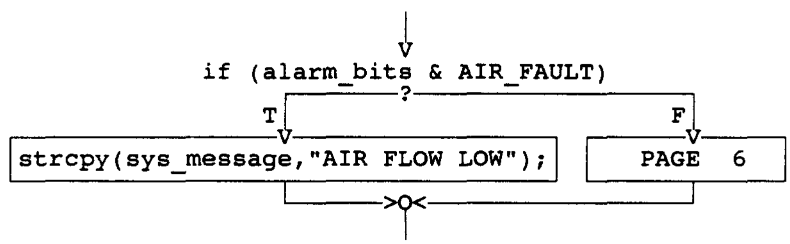

if (avgs < 1)
?
T
F
avgs = 1;
for (i=0;i<avgs;i++)
?
T
F
pressure += analog_in(BKPRESS_IN);
bkpress = sys.bp_disp_gain*pressure/(float)avgs...
if (bkpress>0.1)
?
T
F
return(bkpress);
return(0.1);

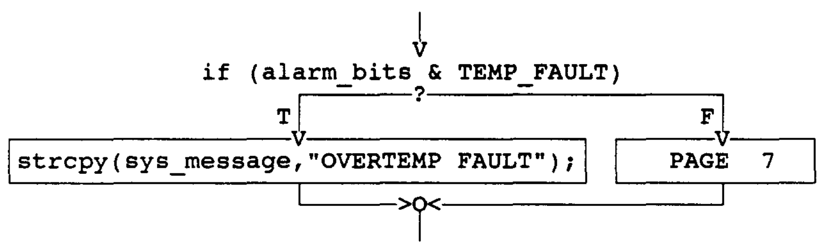

PAGE 2
if (alarm_bits == last_alarm_bits)
?
T
F
if (++seq_fault_count > 10)
last_alarm_bits = alarm_bits;
?
T
F
PAGE 3

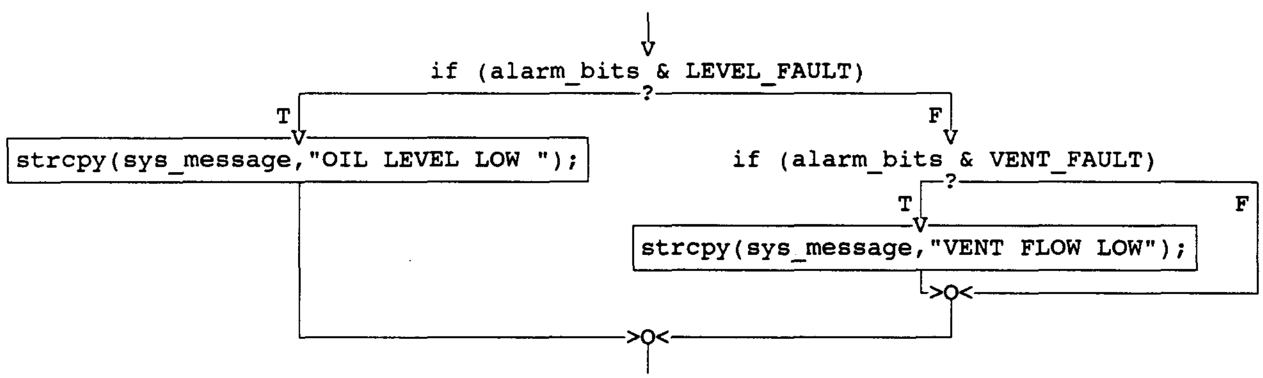

if (++horn_count > 1200)
T
F
horn_count = 0;
io_output(ALARM_LIGHT,OFF);
io_output(ALARM_HORN,OFF);
if (alarm || horn)
if (horn_count == 600)
if (horn)
io_output(ALARM_HORN,ON);
if (alarm)
io_output(ALARM_LIGHT,ON);
alarm_bits = alarm_bits & ALARM_MASK; /*** mas...
if (gpib_fault)
alarm_bits = alarm_bits | GPIB_FAULT_BIT; /*** ...
if (alarm_bits == working_alarm_bits)
return(0);

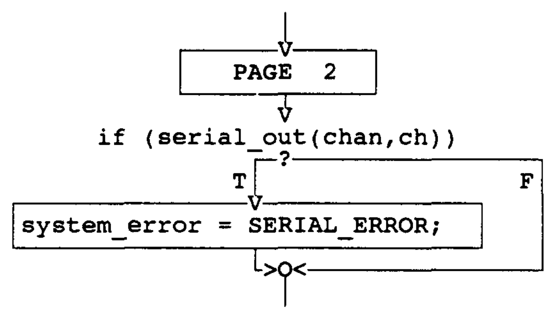

seq_fault_count = 0;
working_alarm_bits = alarm_bits;
if (!alarm_bits)
T
F
horn = alarm = OFF;
if (alarm_bits & GPIB_FAULT_BIT)
T
F
; /* sys_message = previous gpib message */
PAGE 4
horn = alarm = ON;
printf("%s\n",sys_message);
#endif

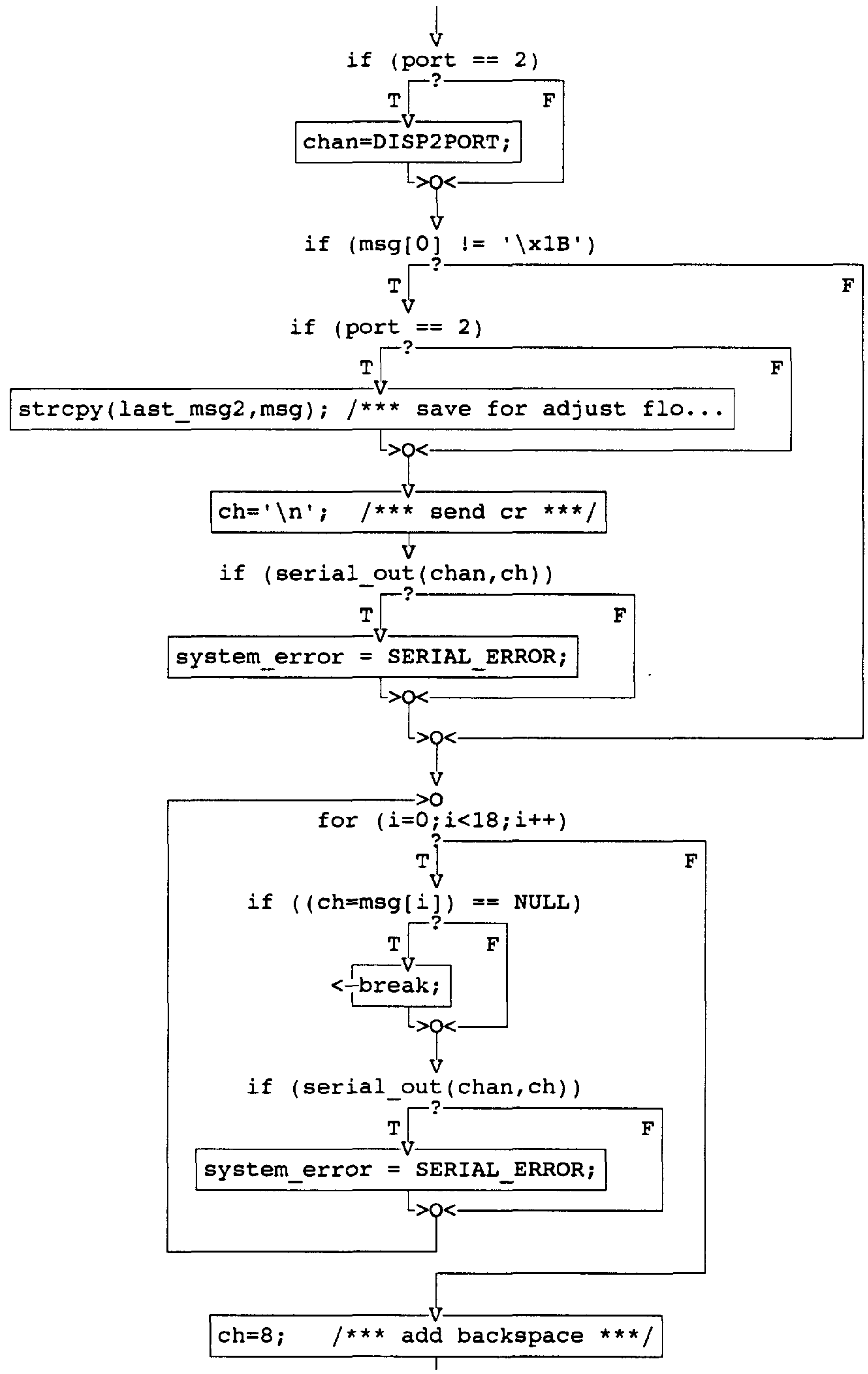

if (alarm_bits & WATER_FAULT)
?
T
F
strcpy(sys_message,"WATER PRESSR LOW");
PAGE 5


if (alarm_bits & AIR_FAULT)
?
T
F
strcpy(sys_message,"AIR FLOW LOW");
PAGE 6


if (alarm_bits & TEMP_FAULT)
?
T
F
strcpy(sys_message,"OVERTEMP FAULT");
PAGE 7


if (alarm_bits & LEVEL_FAULT)
T
F
strcpy(sys_message,"OIL LEVEL LOW ");
if (alarm_bits & VENT_FAULT)
T
F
strcpy(sys_message,"VENT FLOW LOW");

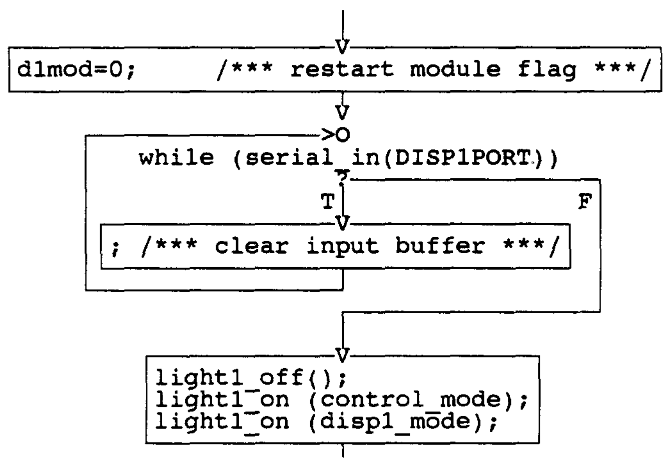

PAGE 2
if (serial_out(chan,ch))
?
T
F
system_error = SERIAL_ERROR;

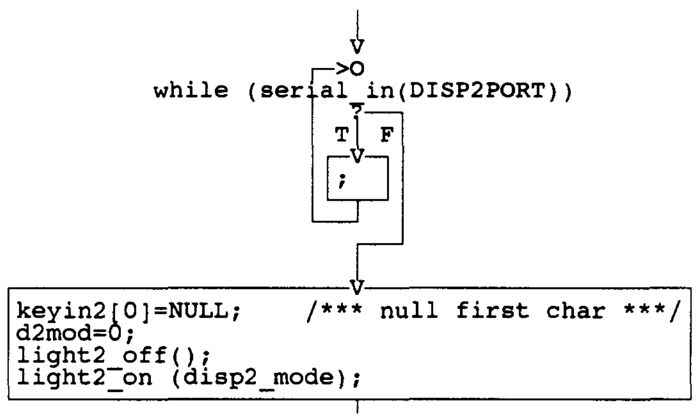

if (port == 2)
T
F
chan=DISP2PORT;
if (msg[0] != '\x1B')
T
F
if (port == 2)
T
F
strcpy(last_msg2,msg); /*** save for adjust flo...
ch='\n'; /*** send cr ***/
if (serial_out(chan,ch))
T
F
system_error = SERIAL_ERROR;
for (i=0;i<18;i++)
T
F
if ((ch=msg[i]) == NULL)
T
F
break;
if (serial_out(chan,ch))
T
F
system_error = SERIAL_ERROR;
ch=8;   /*** add backspace ***/

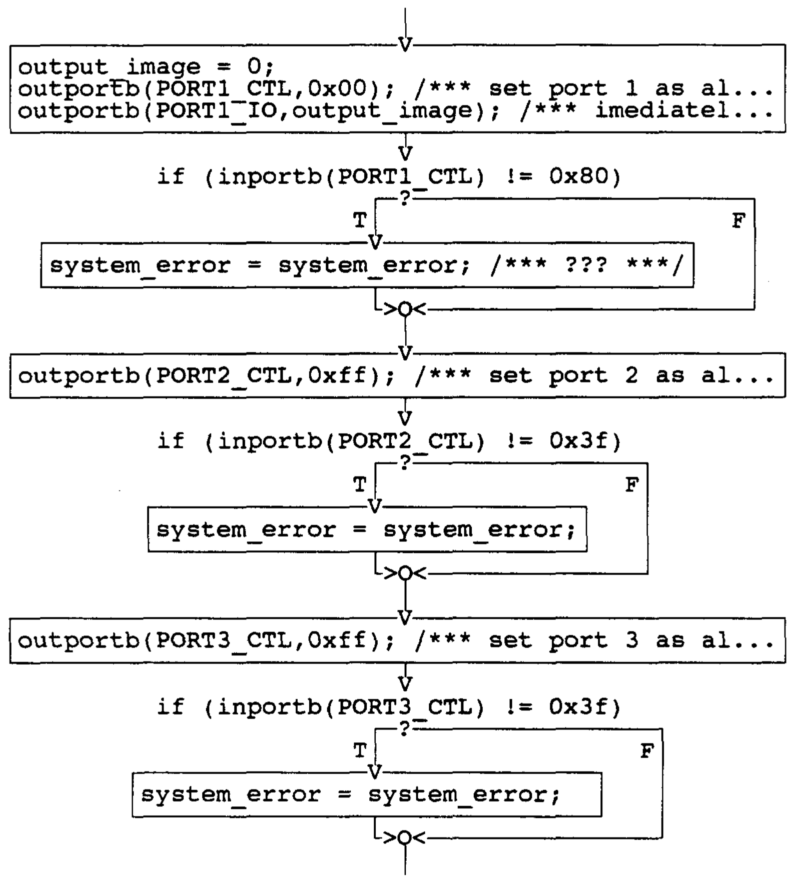
```
strcpy(string,LIGHTONCMD);
number -= CONVRTO_LT;
string[2]=(char)number;
disp(1,string);
```

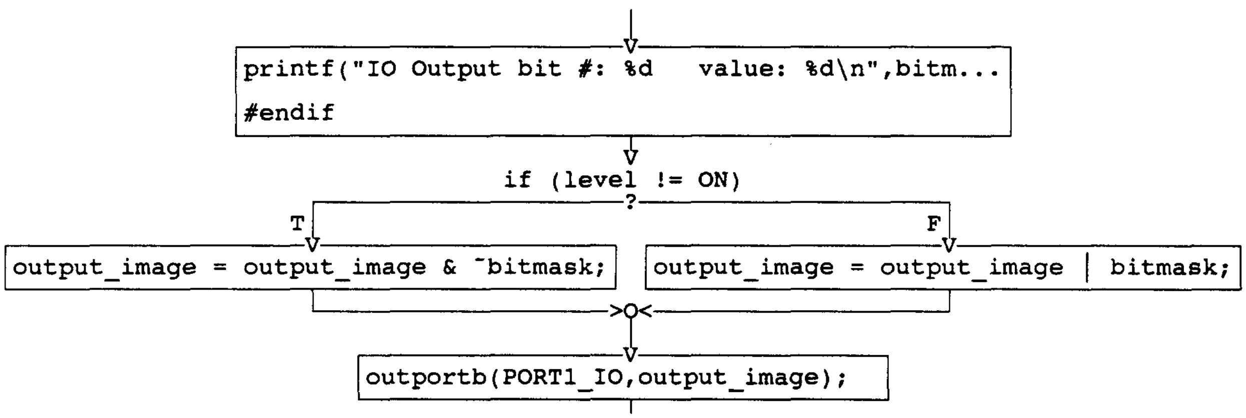
```
strcpy(string,LIGHTONCMD);
number -= CONVRTO_LT;
string[2]=number;
disp(2,string);
```

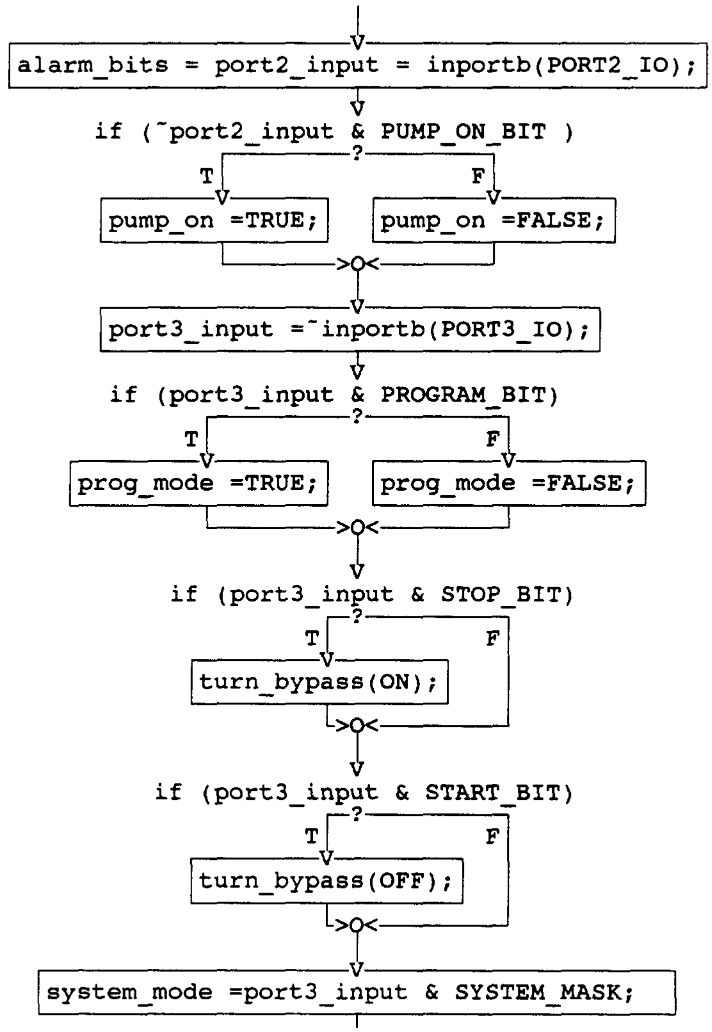
```
disp(2,LIGHTOFFCMD);
```


d1mod=0; /*** restart module flag ***/
while (serial_in(DISP1PORT))
T
F
; /*** clear input buffer ***/
light1_off();
light1_on (control_mode);
light1_on (disp1_mode);

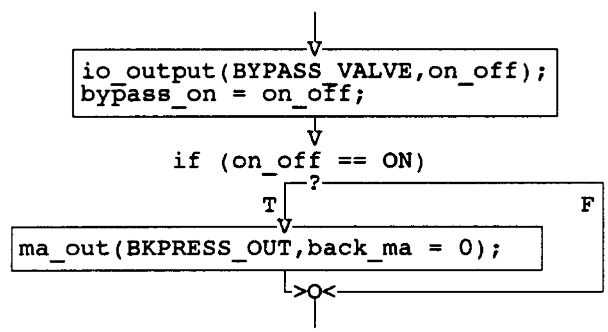

while (serial_in(DISP2PORT))
T
F
;
keyin2[0]=NULL;   /*** null first char ***/
d2mod=0;
light2_off();
light2_on (disp2_mode);

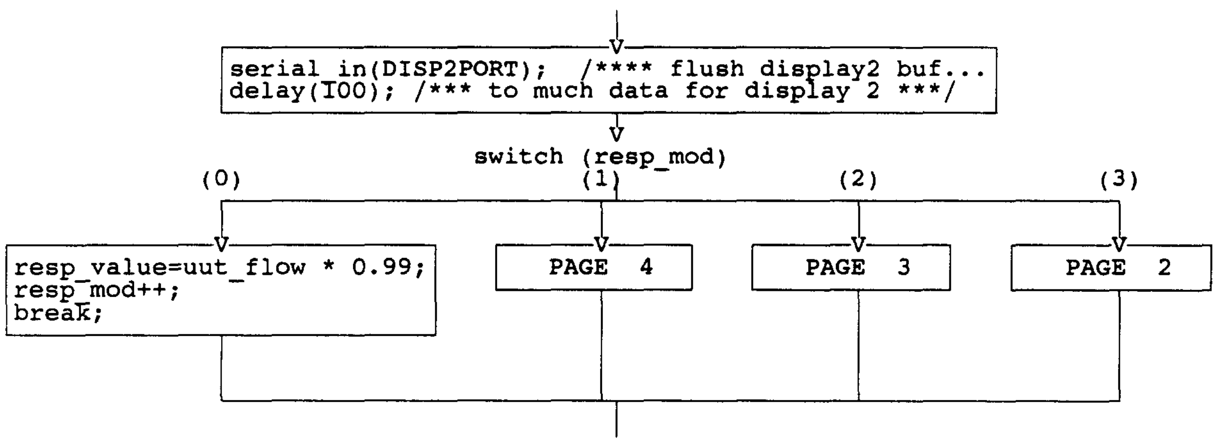

output_image = 0;
outportb(PORT1_CTL,0x00); /*** set port 1 as al...
outportb(PORT1_IO,output_image); /*** imediatel...
if (inportb(PORT1_CTL) != 0x80)
?
T
F
system_error = system_error; /*** ??? ***/
outportb(PORT2_CTL,0xff); /*** set port 2 as al...
if (inportb(PORT2_CTL) != 0x3f)
?
T
F
system_error = system_error;
outportb(PORT3_CTL,0xff); /*** set port 3 as al...
if (inportb(PORT3_CTL) != 0x3f)
?
T
F
system_error = system_error;

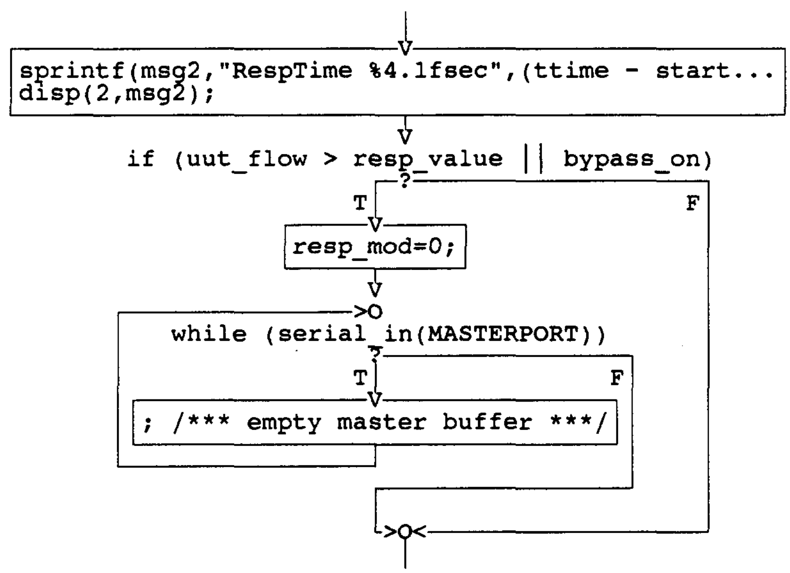

printf("IO Output bit #: %d   value: %d\n",bitm...
#endif
if (level != ON)
T
F
output_image = output_image & ~bitmask;
output_image = output_image | bitmask;
outportb(PORT1_IO,output_image);

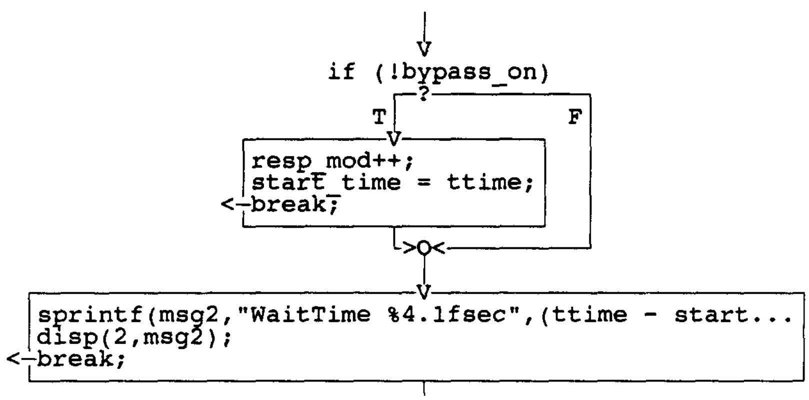

alarm_bits = port2_input = inportb(PORT2_IO);
if (~port2_input & PUMP_ON_BIT)
T
F
pump_on =TRUE;
pump_on =FALSE;
port3_input =~inportb(PORT3_IO);
if (port3_input & PROGRAM_BIT)
T
F
prog_mode =TRUE;
prog_mode =FALSE;
if (port3_input & STOP_BIT)
T
F
turn_bypass(ON);
if (port3_input & START_BIT)
T
F
turn_bypass(OFF);
system_mode =port3_input & SYSTEM_MASK;

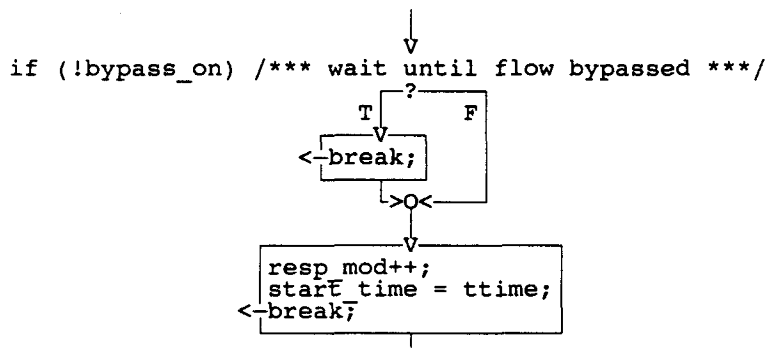
```
io_output(MASTER_VALVE,HIGH);
io_output(HI_LO_RELAY,HIGH);
mastermeter = HIGH;
```


io_output(BYPASS_VALVE,on_off);
bypass_on = on_off;
if (on_off == ON)
?
T
F
ma_out(BKPRESS_OUT,back_ma = 0);


serial_in(DISP2PORT); /**** flush display2 buf...
delay(100); /*** to much data for display 2 ***/
switch (resp_mod)
(0)
(1)
(2)
(3)
resp_value=uut_flow * 0.99;
resp_mod++;
break;
PAGE 4
PAGE 3
PAGE 2

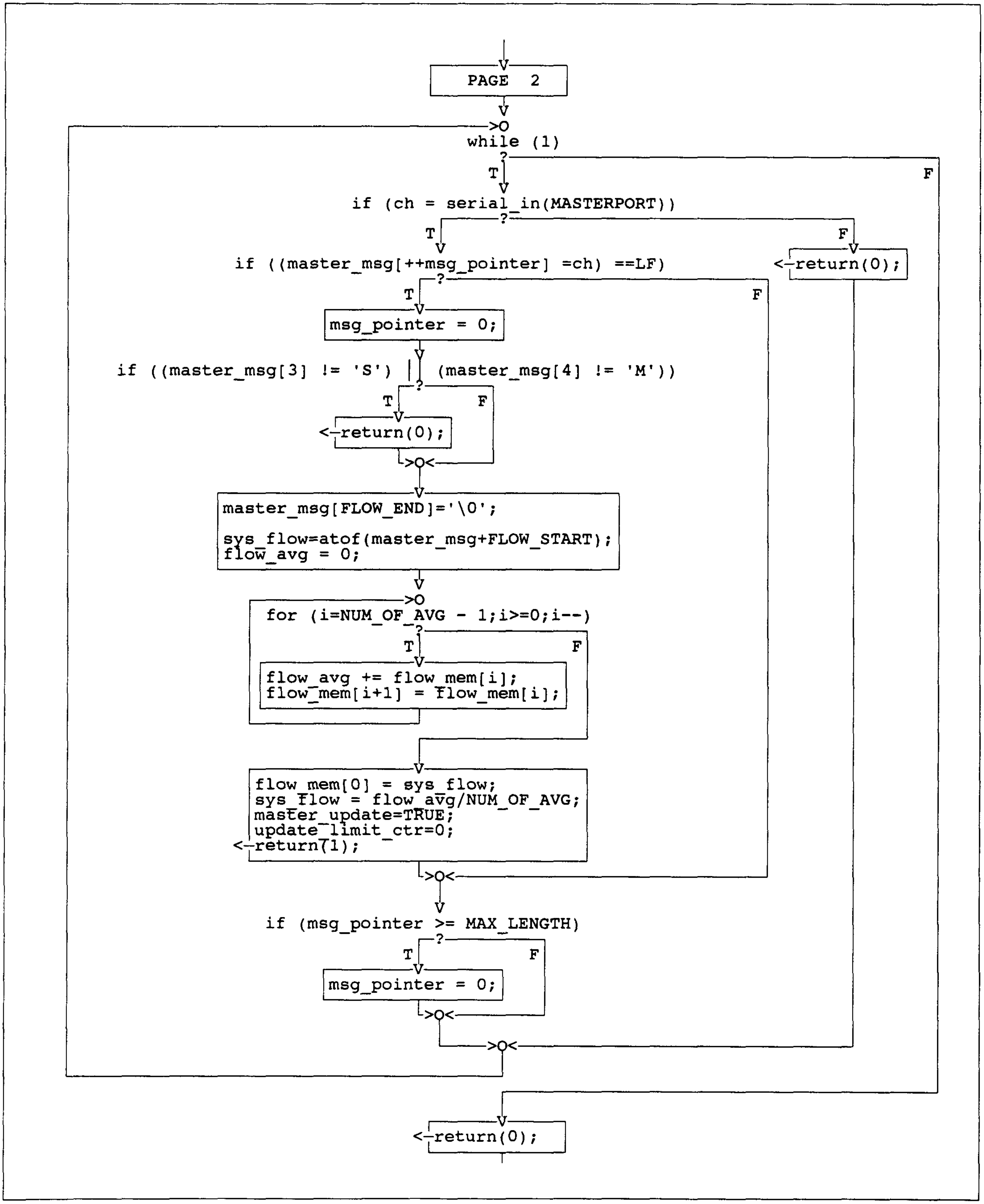

sprintf(msg2,"RespTime %4.1fsec",(ttime - start...
disp(2,msg2);
if (uut_flow > resp_value || bypass_on)
T
F
resp_mod=0;
while (serial_in(MASTERPORT))
T
F
; /*** empty master buffer ***/


if (!bypass_on)
?
T
F
resp_mod++;
start_time = ttime;
<-break;
sprintf(msg2,"WaitTime %4.1fsec",(ttime - start...
disp(2,msg2);
<-break;

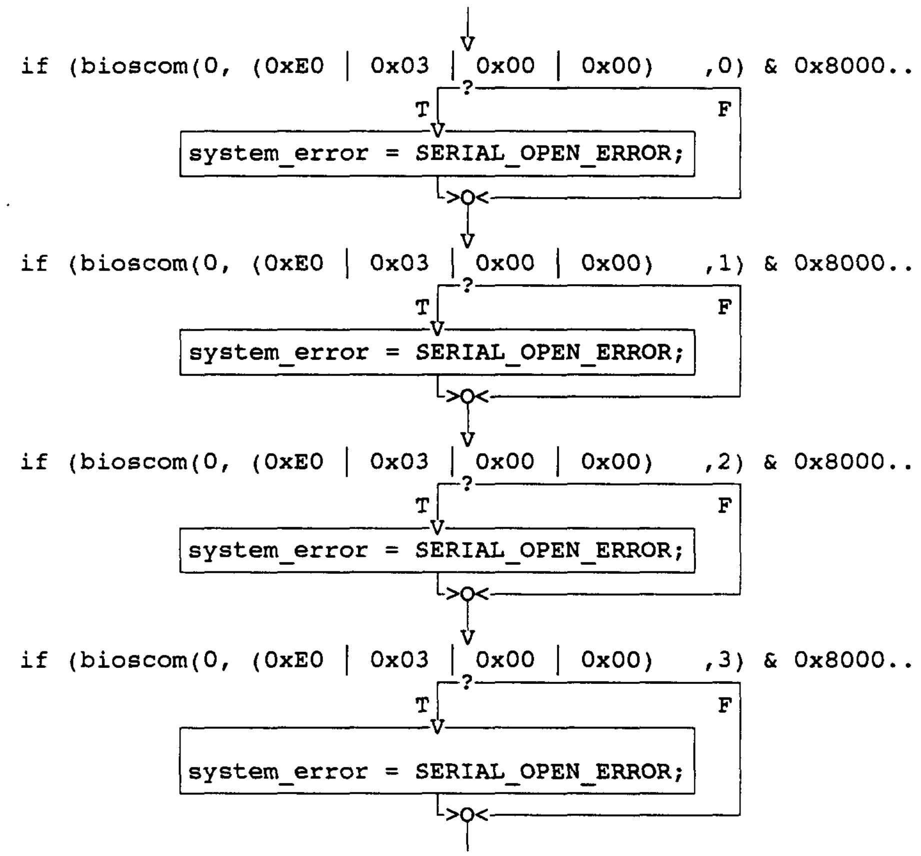

if (!bypass_on) /*** wait until flow bypassed ***/
?
T
F
<-break;
>O<
resp_mod++;
start_time = ttime;
<-break;

```
disp(1,LIGHTOFFCMD);
```

```
io_output(MASTER_VALVE,LOW);
io_output(HI_LO_RELAY,LOW);
mastermeter = LOW;
```


PAGE 2
while (1)
T
F
if (ch = serial_in(MASTERPORT))
T
F
if ((master_msg[++msg_pointer] =ch) ==LF)
return(0);
T
F
msg_pointer = 0;
if ((master_msg[3] != 'S') || (master_msg[4] != 'M'))
T
F
return(0);
master_msg[FLOW_END]='\0';
sys_flow=atof(master_msg+FLOW_START);
flow_avg = 0;
for (i=NUM_OF_AVG - 1;i>=0;i--)
T
F
flow_avg += flow_mem[i];
flow_mem[i+1] = flow_mem[i];
flow_mem[0] = sys_flow;
sys_flow = flow_avg/NUM_OF_AVG;
master_update=TRUE;
update_limit_ctr=0;
return(1);
if (msg_pointer >= MAX_LENGTH)
T
F
msg_pointer = 0;
return(0);

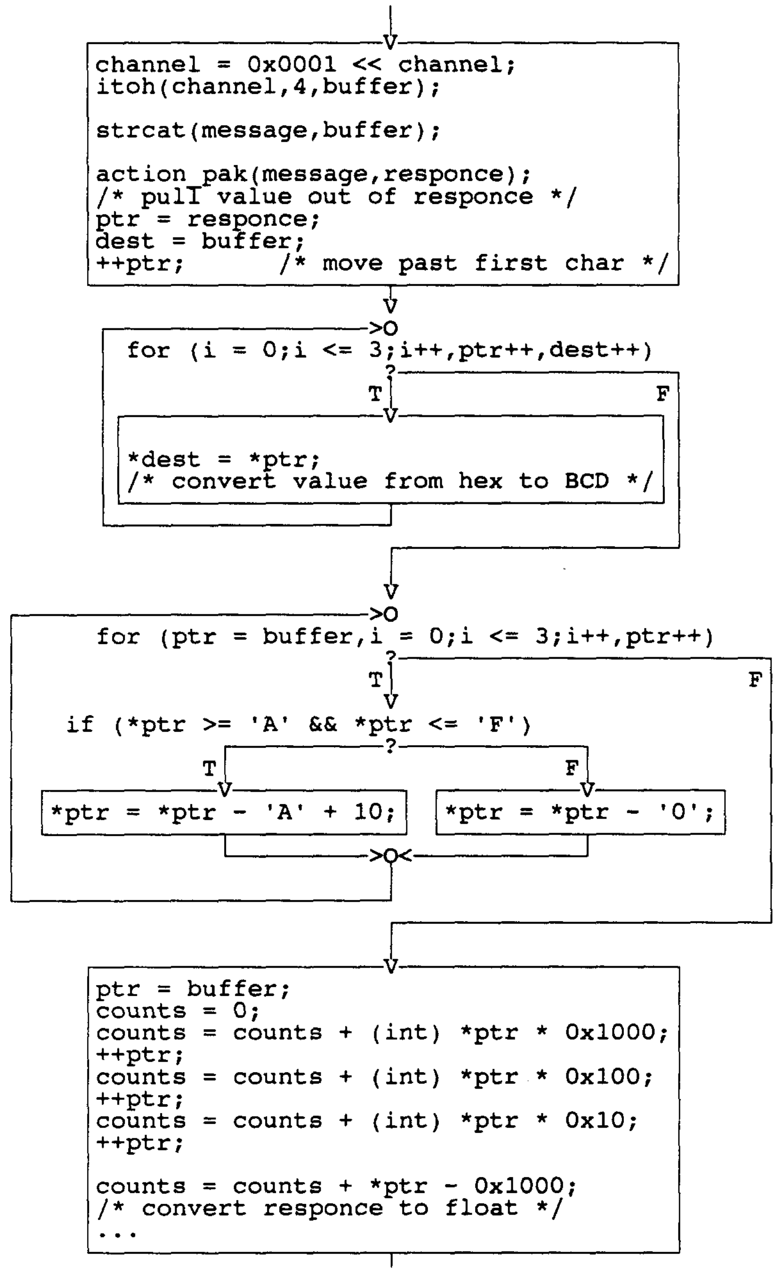
```
if (++update_limit_ctr>10000)
T                                    F
strcpy(sys_message,"MASTER FLOW Err");
gpib_fault=TRUE;
```

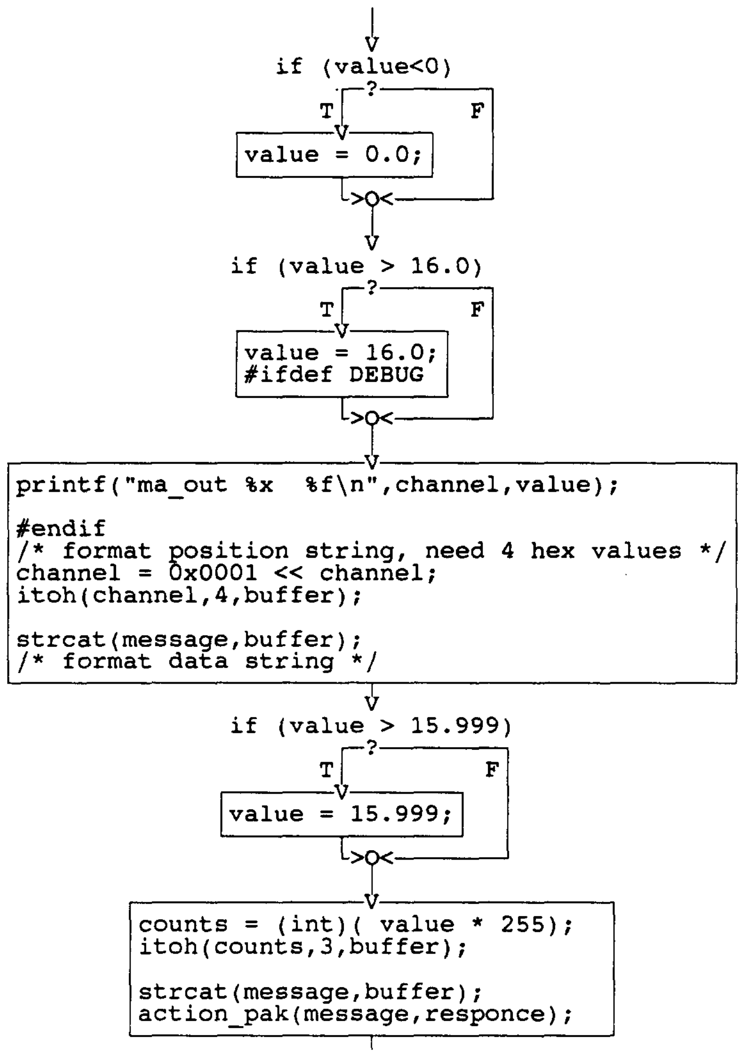

if (bioscom(0, (0xE0 | 0x03 | 0x00 | 0x00) ,0) & 0x8000...
?
T
F
system_error = SERIAL_OPEN_ERROR;
if (bioscom(0, (0xE0 | 0x03 | 0x00 | 0x00) ,1) & 0x8000...
?
T
F
system_error = SERIAL_OPEN_ERROR;
if (bioscom(0, (0xE0 | 0x03 | 0x00 | 0x00) ,2) & 0x8000...
?
T
F
system_error = SERIAL_OPEN_ERROR;
if (bioscom(0, (0xE0 | 0x03 | 0x00 | 0x00) ,3) & 0x8000...
?
T
F
system_error = SERIAL_OPEN_ERROR;

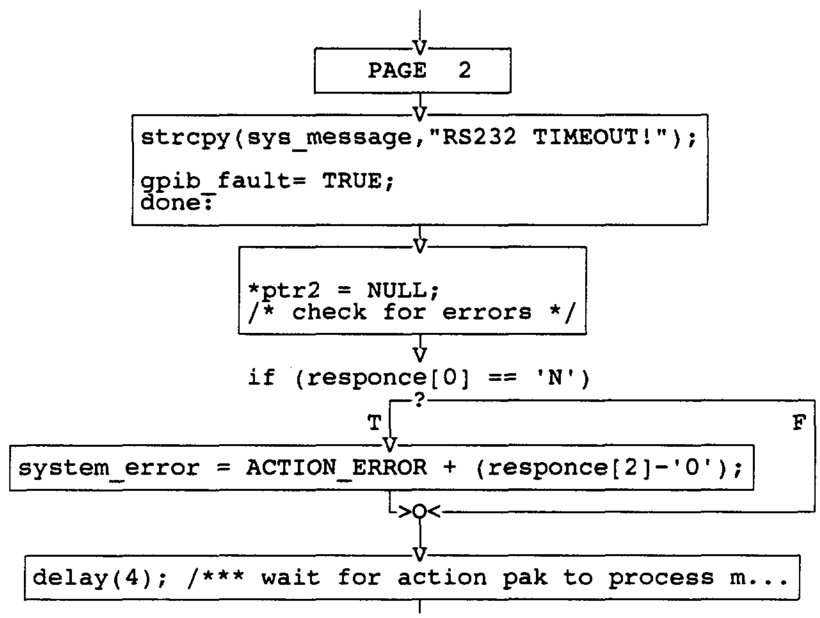
```
                                  |
                                  V
if (bioscom(3,0,chan) & 0x100)   /*** if DATA READY ***/
                               ┌─?─────────────────────┐
                              T│                       │F
                    ┌──────────V──────────────┐        │
                  <─┤return(bioscom(2,0,chan));│        │
                    └─────────────────────────┘        │
                               └─>O<───────────────────┘
                                  │
                          ┌───────V──┐
                        <─┤return(0);│
                          └──────────┘
                                  |
```

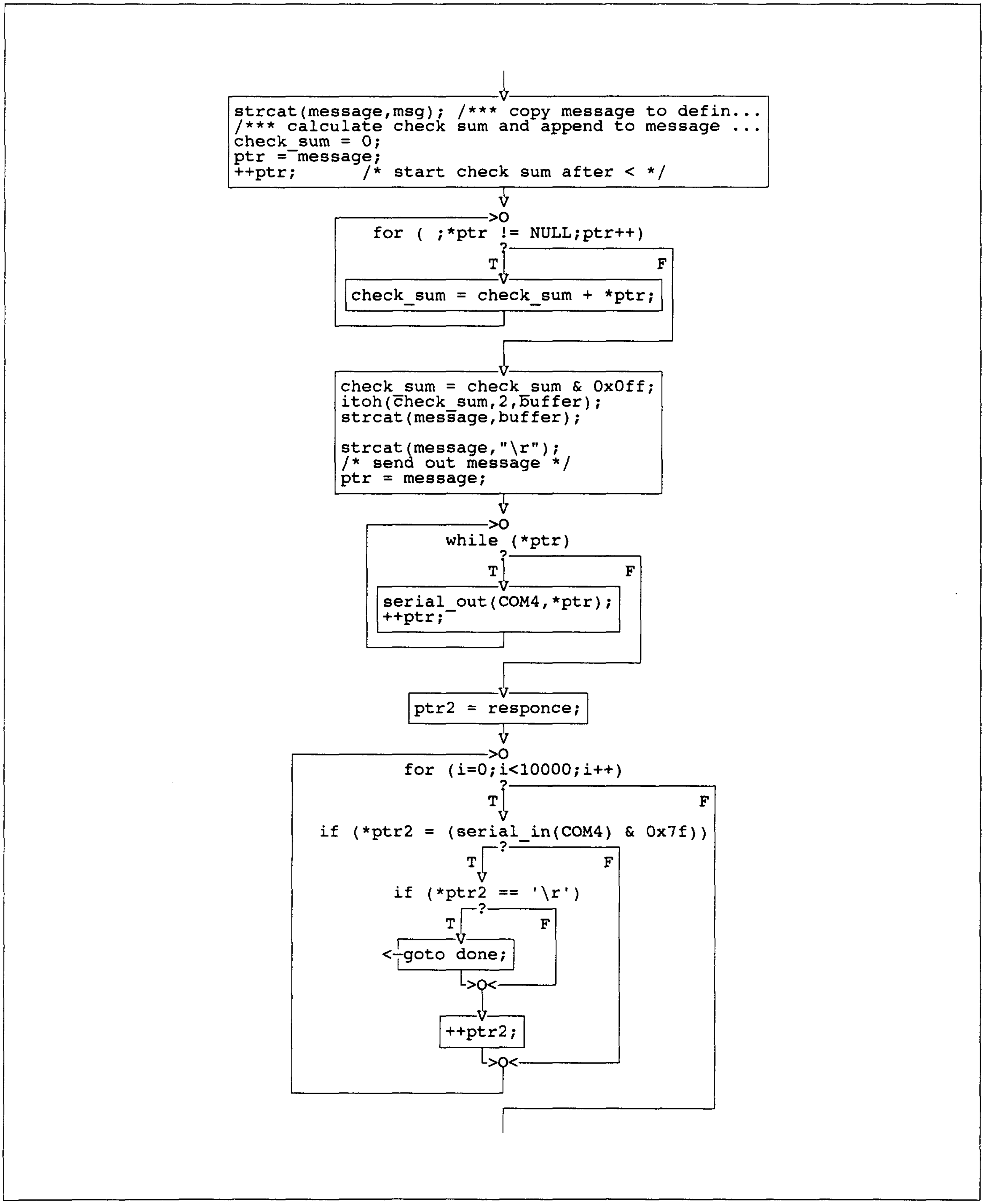
```
                    |
                    V
 |status = bioscom(1,data,chan);
<-return (status & 0x8000);   /*** Mask for TIMEO...
```

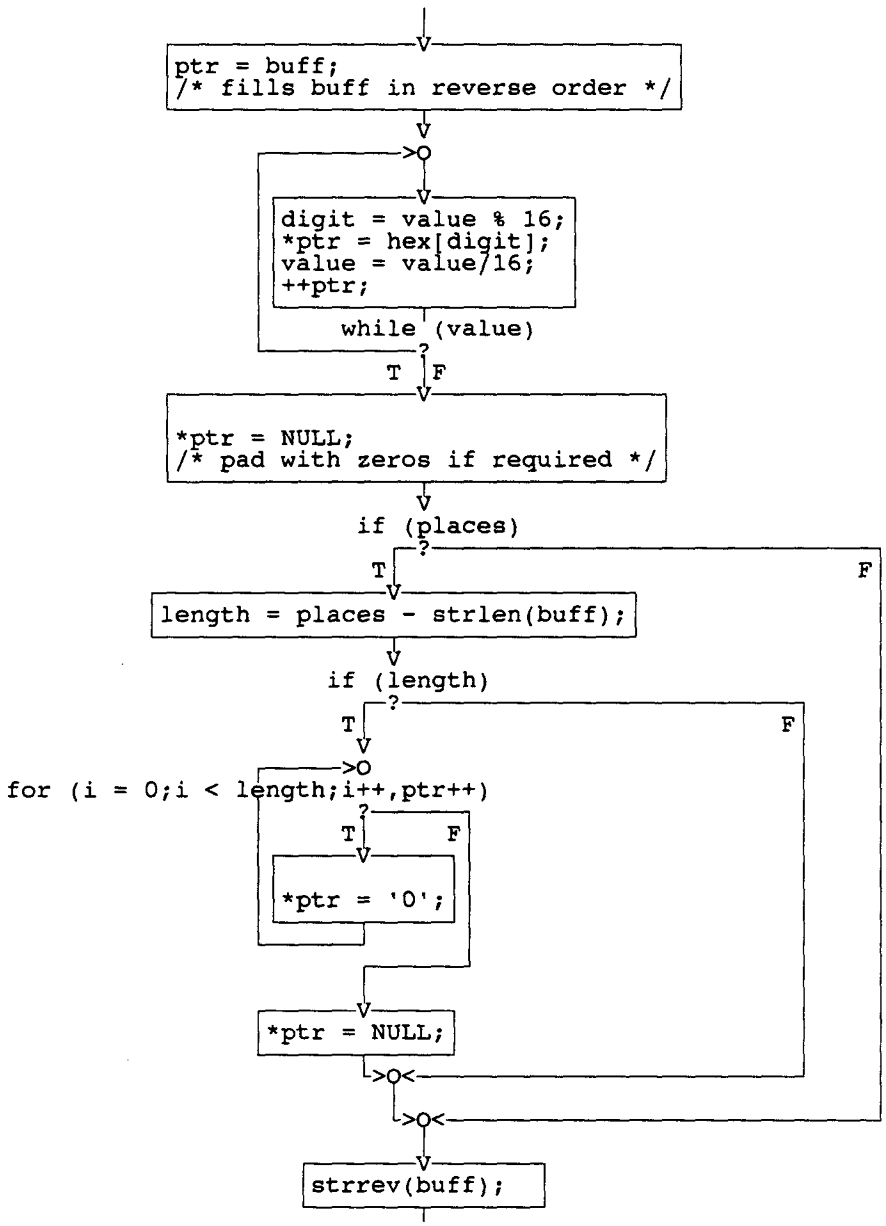
```
action_pak(">04A",responce);
/*** power up clear, digital area ***/

action_pak(">44A",responce);
/*** power up clear, analog area ***/

action_pak(">84A",responce);
/*** configure modules as inputs for pos 14,13,...
using 0111,1101,0100,0000.  Reduce 16 bits to t...
hex chars "7D40", preceded by ">84H" ***/
```


channel = 0x0001 << channel;
itoh(channel,4,buffer);
strcat(message,buffer);
action_pak(message,responce);
/* pull value out of responce */
ptr = responce;
dest = buffer;
++ptr; /* move past first char */
for (i = 0;i <= 3;i++,ptr++,dest++)
?
T
F
*dest = *ptr;
/* convert value from hex to BCD */
for (ptr = buffer,i = 0;i <= 3;i++,ptr++)
?
T
F
if (*ptr >= 'A' && *ptr <= 'F')
?
T
F
*ptr = *ptr - 'A' + 10;
*ptr = *ptr - '0';
ptr = buffer;
counts = 0;
counts = counts + (int) *ptr * 0x1000;
++ptr;
counts = counts + (int) *ptr * 0x100;
++ptr;
counts = counts + (int) *ptr * 0x10;
++ptr;
counts = counts + *ptr - 0x1000;
/* convert responce to float */
...


if (value<0)
?
T
F
value = 0.0;
if (value > 16.0)
?
T
F
value = 16.0;
#ifdef DEBUG
printf("ma_out %x %f\n",channel,value);
#endif
/* format position string, need 4 hex values */
channel = 0x0001 << channel;
itoh(channel,4,buffer);
strcat(message,buffer);
/* format data string */
if (value > 15.999)
?
T
F
value = 15.999;
counts = (int)(value * 255);
itoh(counts,3,buffer);
strcat(message,buffer);
action_pak(message,responce);


PAGE 2
strcpy(sys_message,"RS232 TIMEOUT!");
gpib_fault= TRUE;
done;
*ptr2 = NULL;
/* check for errors */
if (responce[0] == 'N')
?
T
F
system_error = ACTION_ERROR + (responce[2]-'0');
delay(4); /*** wait for action pak to process m...


strcat(message,msg); /*** copy message to defin...
/*** calculate check sum and append to message ...
check_sum = 0;
ptr = message;
++ptr;   /* start check sum after < */
for (;*ptr != NULL;ptr++)
?
T
F
check_sum = check_sum + *ptr;
check_sum = check_sum & 0x0ff;
itoh(check_sum,2,buffer);
strcat(message,buffer);
strcat(message,"\r");
/* send out message */
ptr = message;
while (*ptr)
?
T
F
serial_out(COM4,*ptr);
++ptr;
ptr2 = responce;
for (i=0;i<10000;i++)
?
T
F
if (*ptr2 = (serial_in(COM4) & 0x7f))
?
T
F
if (*ptr2 == '\r')
?
T
F
<-goto done;
++ptr2;


ptr = buff;
/* fills buff in reverse order */
digit = value % 16;
*ptr = hex[digit];
value = value/16;
++ptr;
while (value)
T
F
*ptr = NULL;
/* pad with zeros if required */
if (places)
T
F
length = places - strlen(buff);
if (length)
T
F
for (i = 0;i < length;i++,ptr++)
T
F
*ptr = '0';
*ptr = NULL;
strrev(buff);

```
/***********************************************************
                FUEL FLOW CONTROL PROGRAM FOR RATE OF FLOW TEST STAND
                Original Date: 5/2/89          Programmer:  R. Schilken
                SATE Corporation, Ventura, CA,  under contract to
                APS Systems, Oxnard CA

                                All rights reserved - SATE Corporation
                Compiled with Borland Turbo-C using SMALL model.
                This program requires the following to execute:
                        FUEL.C  This file
                        FUEL.H  Custom header for this program
                        MID.exe executed prior to running, for serial buffers
                        National Insturments GPIB.COM in config.sys and MCIBS.OBJ
                                to link with this object code.

                Ver     Description                                         Date
                ---------------------------------------------------------
                23      Final release with all modifications              5/8/91


        ******************** INCLUDES *****************************/

        #include <stdlib.h>
        #include <bios.h>
        #include <stdio.h>
        #include <string.h>
        #include <math.h>
        #include "fuel.h"       /*** custom header for fuel program ***/

        /********************** MAIN MODULE ***********************/
        main(){
                system_error = FALSE;
```

```
initialize();                   /*** initialize interfaces ***/
cmd_bypass=ON;
bp_enable=OFF;                  /* defeat loop */
bp_val=0;
adjust_flow(1);                 /* set essentially no flow */
displ_mode=SETUP;
disp2_mode=MESSAGES;
sel_mod1=0;                             /* reset uut process # */
sel_mod2=0;                             /* reset select test module */
group=UUT;
newval=0;
prog_mod=0;
test_code=10;
last_sys=99;                    /*** force standby initialize ***/
horn = alarm = OFF;
switch_input();                 /**** test for program mode ****/
master_time=biostime(0,0); /*** dummy read ***/
ma_out(BKPRESS_OUT,back_ma = 0);        /*** force to no backpressure ***/

/********* START OF NORMAL OPERATIONAL CONTROL PROGRAM **********/
while(!system_error || prog_mode){
        ttime = biostime(0,0);          /*** get present dos time ***/
        switch_input();
        alarm_proc();
        if (system_mode ==STANDBY){
                if (system_mode != last_sys){   /*** first time in STANDBY initialize all ***/
                        last_sys = system_mode;
                        if (uut[table_num].max_flow < SMALL_MAX)
                                        master_small();
                        else    master_large();
                        reset_displ();
                        cmd_bypass=ON;
                        control_mode=NORMAL;
                        displ_mode=FLOW;
                        set_mode=FLOWSET;
                        ma_out(BKPRESS_OUT,back_ma = 0); /*** set backpressure to 0 ***/
                        if (flow_value) adjust_flow(0);
                        newval=0;
                        message_mode = TRUE;
                        turn_bypass(ON);
                        horn = OFF;             /*** turn off existing horn ***/
```

```
				disp2_mode=MESSAGES;
				reset_disp2();
				disp(2,sys_message);
			}
			if (program_mode())
				;	/*** then skip select_uut ***/
			else select_uut();
		}
		else{	/*** not STANDBY ***/
			disp1_service();		/*** performed always ***/
			master_flow();  /*** master flow updated ***/
			uut_flowcalc(); /*** update specific uut value **/
			if (system_mode == AUTO){
				set_mode=FLOWSET;	/*** limit to flow only ***/
				disp2_mode=FLOWSET;
			}
			else if (system_mode == EXTERNAL){
				external_control();
				continue;	/*** skip rest of loop ***/
			}
			if (control_mode==RESPONSE){
				response_service();	/*** fixed automatic control routine ***/
				continue;	/*** skip instructions below ***/
			}
			else if (control_mode==FREEZE){ /*** hold valve settings ***/
				master_flow();			/*** flush master flow buffer ***/
				serial_in(DISP2PORT);	/**** flush display2 buffer ***/
			}
			else {			/*** not performed during RESPONSE or FREEZE ***/
				bp_service(PID_LOOP);
				disp2_service();
			}
			if(system_mode!=last_sys){
				reset_disp2();
				reset_disp1();
			}
			last_sys = system_mode;
		}	/*** end else standby ***/
		resp_mod=0;			/*** reset response module ***/
	} /*** endwhile !system_fail ****/
```

```
        /*************  SYSTEM FAIL  ***************/
        adjust_flow(0);
        strcpy(msg1,">>>>SYSTEM FAIL<<<<");
        disp(1,msg1);
        sprintf(msg2,"ERROR CODE %d",system_error);
        disp(2,msg2);
#ifdef DEBUG
        printf("SYSTEM FAILURE CODE: %d\n",system_error);
#endif
        exit (7);
} /*** end main ***/


/*************************************************

                INITIALIZE I/O INTERFACES
        Module used by main to initialize all hardware

**************************************************/
initialize(void){
        FILE *fp;
        int i;
        parallel_io_init();     /*** first initialize parallel io ***/
        serial_init();          /** then serial port & clear displays ***/
        disp(1,"APS SYSTEMS");
        disp(2,"REDNG CNFIG FILE");
        /******* read uut & system data from disk **********/
        if (fp =fopen("UUT.DAT","rb") ){
                for(i=0;i<MAX_UUTS;i++){
                        if(fread(&uut[i],sizeof(struct uut_data),1,fp) !=1)
                                system_error = FILE_READ_ERROR;
                }
                if(     fclose(fp))
                        system_error = FILE_CLOSE_ERROR;
        }
        else return (system_error = FILE_OPEN_ERROR);

        if (fp =fopen("SYS.DAT","rb") ){
                if(fread(&sys,sizeof(struct sys_table),1,fp) !=1)
                        system_error = FILE_READ_ERROR;
                if(     fclose(fp))
```

```
                        system_error = FILE_CLOSE_ERROR;
        }
        else return (system_error = FILE_OPEN_ERROR);

        if (fp =fopen("CAL.DAT","rb") ){
                if(fread(&cal_up,sizeof(cal_up),1,fp) !=1)
                        system_error = FILE_READ_ERROR;
                if(fread(&cal_down,sizeof(cal_down),1,fp) !=1)
                        system_error = FILE_READ_ERROR;
                if(fread(&cal_back,sizeof(cal_back),1,fp) !=1)
                        system_error = FILE_READ_ERROR;
                if(     fclose(fp))
                        system_error = FILE_CLOSE_ERROR;
        }
        else return (system_error = FILE_OPEN_ERROR);
#ifdef DEBUG
        for (i=0;i<17;i++){
                printf("ma=%d     %f      %f       %f\n",i,cal_up[0][i],cal_up[1][i],cal_up[2][i]);
        }
#endif
        lin_calc();             /** calc flow linearization based on 1st uut data table **/

        /********  initialize & verify each GPIB port  *********/
        if ((hp = ibfind ("HP5334A")) < 0)              /*** HP5334 timer counter ***/
                system_error = GPIB_INIT_ERROR;
        if (ibclr (hp) & ERR)
                system_error = GPIB_CLEAR_ERROR;
        ibwrt (hp,"IN AU0 CO0 SE1 AS1 AX1 BS1 BX1 TR0 FN4",38);
        if (ibsta & ERR){
                gpib_fault = TRUE;
                strcpy(sys_message,"GPIB SEND Error");
        }
        if ((na8810 = ibfind ("NA8810")) <0)  /*** North Atlantic 8810 Angle Position Indicator ***/
                system_error = GPIB_INIT_ERROR;
        if (ibclr (na8810) & ERR)
                system_error = GPIB_CLEAR_ERROR;
        analog_init();  /*** initialize action pak analog io ***/
        /*** calculate backpressure period based on program variables entered ***/
        pid_period = (int)(18 * sys.bp_Time);
}  /*** end initialize ***/
```

```
/************************************************

                PROGRAM VARIABLES from keyboard #2
        Module used in main, only if key swtich in program mode

************************************************/

program_mode(void){
        int i,table;
        char charin;

        if (prog_mode){
                if (prog_mod==0){
                        prog_mod=1;
                        group=UUT;
                        element=0;
                        table_num=0;
                        copyto_temp();
                        new_prog_data();
                        password_ok=FALSE;
                        cal_in_process=FALSE;
                        yes_enter=FALSE;
                }
                if (charin = serial_in(DISP2PORT)){
                        switch(charin){
                                case(' '):
                                case('-'):
                                case('1'):case('2'):case('3'):case('4'):case('5'):case('6'):
                                case('7'):case('8'):case('9'):case('0'): case('.'):
                                        for (i=0;i<18;i++){
                                                if (keyin2[i] == NULL){
                                                        keyin2[i]=charin;
                                                        break;
                                        }       }
                                        keyin2[i+1]=0;
                                        disp(2,keyin2);
                                        break;
                                case(DELETE):
                                        for (i=0;i<18;i++){
                                                if (keyin2[i] == NULL){
                                                        if (i) keyin2[i-1]=NULL;
```

```
                                break;
                }       }
                disp(2,keyin2);
                break;
        case(CLEAR):
                keyin2[0]=NULL;
                disp(2,"");
                break;
        case(ENTER):
                for(length=0;length<20;length++){
                        if (keyin2[length]==NULL)
                                break;
                }
                if (length ==0){
                        disp(2,last_msg2);
                break;
                }
                new_prog_data();
                break;
        case(UPARROW):
                data_point+=2;
        case(DOWNARROW):
                if (--data_point < 0)
                        data_point++;
                if (group !=UUT)
                break;
                if (element==12){  /*** if linearization data of UUT, then ***/
                        length=0;  /*** simulate no entry & display table data ***/
                        new_prog_data();
                break;
                }
                if (uut_element_type[element] !=C)
                break;
                for(i=0;i<19;i++){
                        if (keyin2[i]==NULL)
                                break;
                }
                if (i==0){
                        keyin2[0]='A';
                        keyin2[1]=0;
                }
```

```
		else{
			if (charin == UPARROW){
				if ((keyin2[--i]+=1) >'Z'|| (keyin2[i] <'A'))	/*** next high
					keyin2[i]='A';
			}
			else{
				if ((keyin2[--i]-=1) <'A')			/*** next lowe
					keyin2[i]='Z';
			}
		}
			disp(2,keyin2);
			break;
	case(GROUP):
		element =0;
		password_ok =FALSE;
		yes_enter =FALSE;
		reset_disp2();
		if(++group > STOREDATA)  group = UUT;
		switch (group){
			case(UUT):
				new_prog_data();
				break;
			case(SYSTEM):
				disp(1,"SYS Data <ENTER");
				disp(2,"PASSWORD>");
				break;
			case(CALIBRATION):
				disp(1,"CALIBRATION ??");
				disp(2," 1=Yes Enter>");
				break;
			case(STOREDATA):
				disp(1,"STORE DATA ??");
				disp(2," 1=Yes Enter>");
		} /*** end switchgroup ***/
		break;
	case(ITEM):
		length=0;
		switch (group){
			case(UUT):
				if (++element >= UUT_LENGTH -1)
					element=0;
```

```
                                                        new_prog_data();
                                                        break;
                                                case(SYSTEM):
                                                        if (password_ok){
                                                                if (++element > SYS_LENGTH -1)
                                                                        element=0;
                                                        }
                                                        new_prog_data();
                                                        break;
                                                case(CALIBRATION):
                                                case(STOREDATA):
                                                break;
                                        }  /*** endswitch group ***/
                                }  /** endswitch charin ***/
                        } /*** endif charin ***/
                        return(1);
                } /*** endif progswitch ***/
                if (prog_mod){                                          /*** reset only exiting prog ***/
                        disp1_mode=SETUP;
                        reset_disp1();
                        disp2_mode=MESSAGES;
                        reset_disp2();
                        prog_mod=0;
                        last_sys = 99;  /*** force STANDBY initialize ***/
                }
                return(0);
}

/**********************************************

                        NEW_PROG_DATA/
                Module used by program_mode to enter new data

**********************************************/
new_prog_data(void){
        FILE *fp;
        char ch,string[80];
        int i,new_code;
        float deg,angle,pph;

        switch (group){
```

```
case(UUT):
        if (element > 0 && element < 9){
                if (length > uut_msg_length[element]){
                        disp(2,"Msg TOO LONG");
                        reset_disp2();
                        break;
        }       }
        if (element != 12)                              /*** reset conversion table ***/
                data_point=0;
        switch (element){
                case(0):
                        disp(1,"UUT Test CODE");
                        if (new_code = atoi(keyin2)){
                                for (i=0;;){
                                        if (uut[i].data_code==new_code){
                                                table_num=i;
                                                copyto_temp();
                                                sprintf(msg2,"Test CODE: %d",new_code);
                                                disp(2,msg2);
                                break;
                                        }
                                        if (uut[i].data_code==0){
                                                table_num=i;
                                                copyto_temp();
                                                uut_temp.data_code=new_code;
                                                sprintf(msg2,"NEW CODE> %d",new_code);
                                                disp(2,msg2);
                                break;
                                        }
                                        if (++i ==MAX_UUTS){
                                                strcpy(last_msg2,"MEMORY FULL!!");
                                                disp(2,last_msg2);
                                break;
                                        }
                                }
                        }
                        else
                           sprintf(msg2,"Enter New> %d",uut[table_num].data_code);
                           disp(2,msg2);
                        break;
                case(1):
```

```
        if (length)
                strcpy(uut_temp.test_order,keyin2);
        disp(2,uut_temp.test_order);
        disp(1,"UUT Type Numbr");
        break;

case(2):
        if (length)
                strcpy(uut_temp.adaptor_num,keyin2);
        disp(2,uut_temp.adaptor_num);
        disp(1,"UUT Adaptor Cable");
        break;

case(3):
        if (length)
                strcpy(uut_temp.volts_400hz,keyin2);
        disp(2,uut_temp.volts_400hz);
        disp(1,"UUT High Volts");
        break;

case(4):
        if (length)
                strcpy(uut_temp.volts_low,keyin2);
        disp(2,uut_temp.volts_low);
        disp(1,"UUT Low Volts");
        break;

case(5):
        if (length)
                strcpy(uut_temp.freq_low,keyin2);
        disp(2,uut_temp.freq_low);
        disp(1,"UUT Low Freq");
        break;

case(6):
        if (length)
                strcpy(uut_temp.phase_low,keyin2);
        disp(2,uut_temp.phase_low);
        disp(1,"UUT Low Phase");
        break;
```

```
case(7):
        if (length)
                strcpy(uut_temp.vib_level,keyin2);
        disp(2,uut_temp.vib_level);
        disp(1,"UUT Vib Level");
        break;

case(8):
        if (length)
                strcpy(uut_temp.temp_set,keyin2);
        disp(2,uut_temp.temp_set);
        disp(1,"UUT Fluid Temp");
        break;

case(9):
        if (length)
                uut_temp.max_flow=atof(keyin2);
        gcvt(uut_temp.max_flow,9,msg2);
        disp(2,msg2);
        disp(1,"UUT Max Flow");
        break;

case(10):
        if (length)
                uut_temp.back_pressure=atof(keyin2);
        gcvt(uut_temp.back_pressure,9,msg2);
        disp(2,msg2);
        disp(1,"UUT Backpress");
        break;

case(11):
        if (length)
                uut_temp.ind_type=atoi(keyin2);
        itoa(uut_temp.ind_type,msg2,10);
        disp(2,msg2);
        disp(1,"UUT Indictr Type");
        break;

case(12):
        if (data_point/2 >= MAX_BREAKPOINTS ){
                disp(2,"BRKPT TABLE FULL");
```

```
                        break;
                        }
                        if (data_point % 2){            /*** if odd ***/
                                deg=atof(keyin2);
                                ch='A';
                                if (length)
                                        uut_temp.angle[data_point/2]=deg;
                                 sprintf(msg2,"ANGL %7.3f deg",uut_temp.angle[data_point/2]);
                        }
                        else{                                           /*** if even ***/
                                pph=atof(keyin2);
                                ch='F';
                                if (length)
                                        uut_temp.pph[data_point/2]=pph;
                                sprintf(msg2,"FLOW %7.0f pph",uut_temp.pph[data_point/2]);
                        }
                        disp(2,msg2);
                        sprintf(msg1,"UUT Cnvrt #%2.0d %c",data_point/2,ch);
                        disp(1,msg1);

                } /*** endswitch element ***/
                break;  /*** uut ***/
        case(SYSTEM):
                if (password_ok){
                        if (length)
                                sys_temp[element]=atof(keyin2);
                        disp(1,sys_msg_table[element]);
                        gcvt(sys_temp[element],9,string);
                        string[17]=0;                   /*** limit length ***/
                        disp(2,string);
                }
                else if (stricmp(keyin2,password) ==0){
                        password_ok=TRUE;
                        disp(1,sys_msg_table[element =0 ]);
                        disp(1,sys_msg_table[element]);
                        gcvt(sys_temp[element],9,string);
                        string[17]=0;                   /*** limit length ***/
                        disp(2,string);
                }
                else{
                        disp(1,"SYS Data <ENTER");
```

```
				disp(2,"PASSWORD>");
			}
			break;

		case(CALIBRATION):
			if (stricmp(keyin2,"1") ==0){
				disp(1,"CAL IN PROCESS");
				if (cal_flow() == FAULT)
					disp(1,">UNSUCCESSFUL<!!");
				else {
				disp(1,"FLOWCAL COMPLETE");
				disp(2,"STOREdata Next!!");
			}	}
			else disp(2,"???");
			break;
		case(STOREDATA):
			disp(2,"Working . . .");
			copyto_mem();
			system("DEL UUT.BAK");
			system("DEL SYS.BAK");
			system("DEL CAL.BAK");
			system("REN UUT.DAT UUT.BAK");
			system("REN SYS.DAT SYS.BAK");
			system("REN CAL.DAT CAL.BAK");
			if (fp =fopen("UUT.DAT","wb") ){
				for(i=0;i<MAX_UUTS;i++){
					if(fwrite(&uut[i],sizeof(struct uut_data),1,fp) !=1)
						system_error = FILE_WRITE_ERROR;
				}
				if(	fclose(fp))
					system_error = FILE_WRITE_ERROR;
			}
			else  system_error = FILE_OPEN_ERROR;

			if (fp =fopen("SYS.DAT","wb") ){
				if(fwrite(&sys,sizeof(struct sys_table),1,fp) !=1)
					system_error = FILE_WRITE_ERROR;
				if(	fclose(fp))
					system_error = FILE_WRITE_ERROR;
			}
			else  system_error = FILE_OPEN_ERROR;
```

```
                if (fp =fopen("CAL.DAT","wb") ){
                        if(fwrite(&cal_up,sizeof(cal_up),1,fp) !=1)
                                system_error = FILE_WRITE_ERROR;
                        if(fwrite(&cal_down,sizeof(cal_down),1,fp) !=1)
                                system_error = FILE_WRITE_ERROR;
                        if(fwrite(&cal_back,sizeof(cal_back),1,fp) !=1)
                                system_error = FILE_WRITE_ERROR;
                        if(     fclose(fp))
                                system_error = FILE_WRITE_ERROR;
                }
                else  system_error = FILE_OPEN_ERROR;

                disp(2,"  Completed!!");
                pid_period = (int)(18 * sys.bp_Time);
                break;

        } /*** endswitch group ***/
        reset_disp2();
}

/**********************************************

                SELECT UUT DATA TABLE
        Module used by main to select new UUT type from keypad #2.

**********************************************/
select_uut(void){
        char charin;
        int i,new_code;
        if (sel_mod2==0){                                       /*** First time only ***/
                disp2_mode=SELECT;
                reset_disp2();
                disp(2,">Enter TEST CODE");
                sel_mod2++;
                sel_mod1=1;                     /*** force display update ***/
                element=0;
                message_mode = FALSE;
        }
        else if (sel_mod2==1){                          /*** Input character ***/
                if (charin = serial_in(DISP2PORT)){
```

```
                if (charin == 'E')
                        last_sys = 99;  /*** force system message ***/
                else if (charin == 'F')
                        sel_mod2 = 0;   /*** force test code entry ***/
                else if (charin == ENTER)
                        sel_mod2++;
                else if (charin == DELETE){
                        for(i=0;i<18;i++){
                                if (keyin2[i]==NULL){
                                        if (i > 0)  keyin2[i-1]=NULL;
                                }
                        }
                        keyin2[i]=NULL;
                }
                else if ((charin == CLEAR) & message_mode){
                        strcpy (sys_message,"  SYSTEM OK");
                        gpib_fault = 0;
                        last_sys = 99;  /*** force system message update ***/
                        horn = OFF;
                }
                else if ((charin>='0' && charin<='9') || charin=='.'){
                        for (i=0;i<18;i++){
                                if (keyin2[i]==NULL){
                                        keyin2[i]=charin;
                                break;
                        }       }
                        keyin2[i+1]=NULL;
                }
                else    sel_mod2++;             /*** restart input ***/
                disp(2,keyin2);         /*** update display with this char ***/
                light2_off();
                light2_on (disp2_mode = SELECT);
        }
}
else{                           /*** Compile input string ***/
        sel_mod2=1;                     /*** for next string ***/
        strcpy(msg2,">CODE Not Found<");        /*** assume bad entry ***/

        if (new_code = atoi(keyin2)){
                for (i=0;i<MAX_UUTS;i++){
                        if (uut[i].data_code==new_code){
```

```
                                test_code=new_code;
                                table_num=i;
                                strcpy(msg2,"Test CODE: ");
                                strcat(msg2,keyin2);
                                sel_mod1++;             /*** force display 1 update ***/
                                lin_calc();                     /*** calc new gains ***/
                            if (uut[table_num].max_flow < SMALL_MAX)
                                        master_small();
                            else    master_large();
                                min_uut_angle = uut[table_num].angle[0] - 10.0;
                                break;
                }       }       }
                disp(2,msg2);
                reset_disp2();
        }
        setup_display();                                                /*** display1 setup routine ***/

} /*** end select_uut ***/

/***************************************************
                SETUP DISPLAY
        Used only by UUT SELECT
***************************************************/
setup_display(void){
        char charin,string[18];
        int i;
        if (sel_mod1==0){
                if (charin = serial_in(DISP1PORT)){
                        sel_mod1++;                                     /*** display next loop ***/
                        if (charin == SETUP){
                                if (++element>8)
                                        element=0;              /*** roll back to element 1 ***/
                        }
                        else
                                element=0;                      /*** any other input reset to 1 ***/
                }
        }
        else{
                sel_mod1=0;
                get_setup_msg(element);
                disp(1,msg1);
```

```
        }
}

/*****************************************
                GET SETUP MESSAGE
        Used by setup_display & displ_service
*****************************************/
get_setup_msg(int element){
        char string[18];

                switch(element){
                        case(0):
                                strcpy(msg1,"Test CODE: ");
                                itoa(uut[table_num].data_code,string,10);
                                strcat(msg1,string);
                                break;
                        case(1):
                                strcpy(msg1,"UUT: ");
                                strcat(msg1,uut[table_num].test_order);
                                break;
                        case(2):
                                strcpy(msg1,"ADPTR: ");
                                strcat(msg1,uut[table_num].adaptor_num);
                                break;
                        case(3):
                                strcpy(msg1,"400Hz V: ");
                                strcat(msg1,uut[table_num].volts_400hz);
                                break;
                        case(4):
                                strcpy(msg1,"Low Volts: ");
                                strcat(msg1,uut[table_num].volts_low);
                                break;
                        case(5):
                                strcpy(msg1,"Low FREQ: ");
                                strcat(msg1,uut[table_num].freq_low);
                                break;
                        case(6):
                                strcpy(msg1,"Low PHASE: ");
                                strcat(msg1,uut[table_num].phase_low);
                                break;
                        case(7):
```

```
                                strcpy(msg1,"VIB Lev: ");
                                strcat(msg1,uut[table_num].vib_level);
                                break;
                        case(8):
                                strcpy(msg1,"TEMP Set: ");
                                strcat(msg1,uut[table_num].temp_set);
                                break;
                }
}


/************************************************
                UUT FLOW LINEARIZATION
        Calculates gain for each line of selected UUT linearization table.
        Called whenever a new xducer is selected.
        Gain =  flow/angle    (between each line and NEXT line in table)
        Note:  First angle in table must be 0; Last angle in table must be
           followed by a zero angle to hault calculations.  Also note
           that the last gain is set to the previous gain for safety.
        Also set master large or small flow meter based on max flow of xducer
*****************************************************/

lin_calc(void){
        int i;
        for (i=1 ; ((i<MAX_BREAKPOINTS)
                && (uut[table_num].angle[i] > uut[table_num].angle[i-1])) ; i++)
                uut_gain[i-1]=(uut[table_num].pph[i]-uut[table_num].pph[i-1])
                        /(uut[table_num].angle[i]-uut[table_num].angle[i-1]);
        uut_gain[i-1]=uut_gain[i-2];
        data_length=i;  /*** save length for run time uut calculations ***/

        if (uut[table_num].max_flow < SMALL_MAX)
                        master_small();
        else    master_large();
}


/****************************************************

                DISPLAY-1 SERVICE
        Call repetedly by main to input characters from
```

```
        keyboard and update display accordingly.
****************************************************/
displ_service(void){
        float power,analog_in();
        char charin,msg[20];

        if (charin = serial_in(DISP1PORT)){
                update_displ=TRUE;
                switch(charin){
                        case(NORMAL):
                        case(FREEZE):
                        case(RESPONSE):
                                control_mode=charin;
                                break;
                        case(FLOW):
                                if (displ_mode == FLOW)
                                        if (++flow_mode > 3)
                                                flow_mode=0;
                                displ_mode=FLOW;
                                break;
                        case(POWER):
                                if (displ_mode == POWER)
                                        if (++power_mode > 1)
                                                power_mode=0;
                                displ_mode=POWER;
                                break;
                        case(SETUP):
                                if (displ_mode==SETUP)
                                        if (++element>8)
                                                element=0;              /*** roll back to element 1 ***/
                                displ_mode=SETUP;
                                break;
                        default:
                                if (displ_mode == SETUP)
                                        element=0;
                } /** end switch **/
                if (displ_mode==SETUP)
                        get_setup_msg(element);
                reset_displ();
        }  /** endif new character **/
        if (update_displ){
```

```
                if (displ_mode == FLOW){
                        if (flow_mode==1)
                                sprintf(msg1,"UUTflo %6.0fpph",uut_flow);
                        else if (flow_mode==2)
                                sprintf(msg1,"MASTER %6.0fpph",sys_flow);
                        else if (flow_mode==3)
                                sprintf(msg1,"DeltaF %6.0fpph",flow_diff);
                        else   /** flow_mode==0) **/
                                sprintf(msg1," Angle %5.2fdeg",uut_angle);

                }
                else if (displ_mode == POWER){
                        if (power_mode){
                                power=(analog_in(HIVOLTS)*analog_in(HIAMPS)*HI_SF);
                                sprintf(msg1,"400Hz: %5.2f VA",power);
                        }
                        else {
                                power=(analog_in(LOWVOLTS)*analog_in(LOWAMPS)*LOW_SF);
                                sprintf(msg1,"LowV:  %5.2f VA",power);
                        }
                }
                disp(1,msg1);                   /*** refresh display ***/
                update_displ=FALSE;
        }
}

/****************************************************

                DISPLAY-2 SERVICE
        Call repetedly by main to input characters from
        keyboard and update display accordingly.
****************************************************/
disp2_service(void){
        int i;
        char charin;
        static float newval;

        if (charin = serial_in(DISP2PORT)){
                switch(charin){
                        case('1'):case('2'):case('3'):case('4'):case('5'):case('6'):
                        case('7'):case('8'):case('9'):case('0'): case('.'):
```

```
		for (i=0;i<18;i++){
			if (keyin2[i] == NULL){
				keyin2[i]=charin;
				break;
		}	}
		keyin2[i+1]=0;
		disp(2,keyin2);
		break;
	case(DELETE):
		for (i=0;i<18;i++){
			if (keyin2[i] == NULL){
				if (i) keyin2[i-1]=NULL;
				break;
		}	}
		disp(2,keyin2);
		break;
	case(CLEAR):
		keyin2[0]=NULL;
		disp(2,keyin2);
		gpib_fault = 0;  /*** if fault message, clear flag ***/
		break;
	case(ENTER):
		newval = atof(keyin2);
		/*** no break; ***/
	case(UPARROW):
	case(DOWNARROW):
		if (disp2_mode == FLOWSET){
			if (charin == UPARROW)
				newval=newval + (0.02*uut[table_num].max_flow);
			else if (charin == DOWNARROW)
				newval=newval - (0.02*uut[table_num].max_flow);
			if (newval < 50)
				newval = 0;
			if (newval > uut[table_num].max_flow)
				newval = uut[table_num].max_flow;
			if(system_mode == MANUAL){
				flow_value = newval;
				bp_service(BP_INIT);
				if (newval > last_pph * 1.1)
					delay(500);		/*** let bp settle ***/
			}
```

```
                adjust_flow(newval);
                sprintf(last_msg2,"Flow: %6.0fpph",flow_value);  /*** save latest entry ***/
            }
            else{ /*** backpressure mode set ***/
                if (charin == UPARROW)
                    newval=newval + 1;
                else if (charin == DOWNARROW)
                    newval=newval - 1;
                if (newval < 0)  newval = 0;
                if (newval > 20)  newval = 20;
                bp_val=(int)newval;
                sprintf(last_msg2,"BkPress: %dpsi",bp_val);  /*** save latest entry ***/
                bp_service(BP_INIT);    /*** set initial backpressure valve ***/
            }
            disp(2,last_msg2);
            reset_disp2();
            break;
        case(FLOWSET):
            disp2_mode = FLOWSET;
            reset_disp2();
            sprintf(last_msg2,"Flow: %6.0fpph",flow_value);  /*** save latest entry ***/
            disp(2,last_msg2);
            newval=flow_value;
            break;
        case (PRESSURESET):
            if (system_mode == MANUAL){
                disp2_mode = PRESSURESET;
                reset_disp2();
                sprintf(last_msg2,"BkPress: %dpsi",bp_val);  /*** save latest entry ***/
                disp(2,last_msg2);
                newval=bp_val;
            }
            break;
        case (MESSAGES):
            disp2_mode = MESSAGES;
            reset_disp2();
            disp(2,sys_message);
            break;

    }  /** endswitch **/
    return(1);
```

```
        }
        return(0);
}

/*******************************************
                UUT FLOW CALCULATION
        Used by main to determine present UUT flow value.
        For each type of uut, communicates with and determines
        the transducer in angle real time.  Task is divided into
        several steps.  Only one step is executed at a time so
        that program will return quickly for other tasks.
*******************************************/
uut_flowcalc(void){
        static int wait_count;
        static long int lasttime;
        static float period,interval;
        char data[50];
        char spr;                       /* serial poll response byte */
        long int thistime,temp;
        static float synch_avg,timed_avg;
        float synch_meas;
        int i;

        switch(uut[table_num].ind_type){
                case(SYNCHRO1): /*** normal ***/
                case(SYNCHRO2): /*** offset by -80 degrees ***/
                        synch_meas = 0;
                        for (i=0;i<2;i++){      /*** average two reads / measurement ***/
                                if (ibrd (na8810,data,8) & ERR){
                                        strcpy(sys_message,"GPIB REC Error");
                                        gpib_fault=TRUE;
                                        break;
                                }
                                data[8]=0;      /*** add string terminator ***/
                                synch_meas +=  atof(data+1);    /*** chars 2 through 7 ***/
                        }
                        if (control_mode==RESPONSE)
                                uut_angle = synch_meas/2000;    /*** single measurement data/1000 ***/
                        else /** average 20 measurements (100 till .01% settle) ***/
                                uut_angle = ((synch_avg=synch_avg-(synch_avg/20)+synch_meas)/40000);
                                                                        /*** remember 20*2000 ***/
```

```
		if (((temp = (thistime = biostime(0,0)) - lasttime)) < 10)	/*** test this against last, a
			if (temp >= 0)		/*** wait .6 sec between display changes ***/
				break;	/*** return ***/
		lasttime = thistime;
		if (uut_angle < min_uut_angle)	/*** determined at start ***/
			uut_angle = uut_angle + 360;
		uut_linearize(uut_angle);
		if (uut[table_num].ind_type == SYNCHRO2)
			if (uut_angle < 0)	/*** display	****/
				uut_angle = uut_angle +360;
		uut_flowmod=0;	/*** reset flowmodule for TIMED ***/
		update_displ=TRUE;
		break;
	case(HARMONIC):
		if (((temp = (thistime = biostime(0,0)) - lasttime)) < 10)	/*** test this against last, a
			if (temp >= 0)	break;	/*** wait .6 sec between reads ***/
		lasttime = thistime;
		/*** process harmonic using 115vac input as reference multiplier ****/
		uut_angle = analog_in(HARMONICCHAN)*analog_in(HIVOLTS)*sys.harmonic_gain
						+ sys.harmonic_offset;
		uut_linearize(uut_angle);
		update_displ=TRUE;
		uut_flowmod=0;	/*** reset flowmodule for TIMED ***/
		break;
	case(TIMED):
		/*** Uses HP5334 timer counter to measure period and time interval between
		* two pulse signals.  Received ascii data string is "SddddddddddddddEee<^"
		* where:  d is signed data;  e is signed exponent;  < is linefeed; ^ is CR ***/
		if (pump_on ==FALSE)
			break;
		switch(uut_flowmod){
			case(0):
				/*** send period request ***/

			    ibwrt (hp,"FN4 WA1",7);
			if (ibsta & ERR){
					strcpy(sys_message,"GPIB SEND Error");
					gpib_fault=TRUE;
				}
				uut_flowmod++;
				wait_count = 0;
```

```
        break;
case(1):
        /*** wait for results ***/
        if (wait_count++ > 200)
                uut_angle = 0;
if (ibrsp (hp,&spr) & ERR){
                strcpy(sys_message,"GPIB TIMEOUT!");
                gpib_fault=TRUE;
        }
        if (spr & 0x01)
                uut_flowmod++;
        break;
case(2):
        /*** get period results ***/
        if (ibrd (hp,data,21) & ERR){
                strcpy(sys_message,"GPIB REC Error");
                gpib_fault=TRUE;
        }
        data[19]=0;     /*** add string terminator at char 19 ***/
        period=atof(data+1);    /*** convert from character 2 through 19 ***/
        uut_flowmod++;
        break;
case(3):
        /*** send time request ***/
        ibwrt (hp,"FN5 WA1",7);
        if (ibsta & ERR){
                strcpy(sys_message,"GPIB SEND Error");
                gpib_fault=TRUE;
        }
        uut_flowmod++;
        wait_count = 0;
        break;
case(4):
        /*** wait for results ***/
        if (wait_count++ > 200)
                uut_angle = 0;
        if (ibrsp (hp,&spr) & ERR){
                strcpy(sys_message,"GPIB TIMEOUT!");
                gpib_fault=TRUE;
        }
        if (spr & 0x01)
```

```
                                        uut_flowmod++;
                                break;
                        case(5):
                                /*** get interval results ***/
                                if (ibrd (hp,data,21) & ERR){
                                        strcpy(sys_message,"GPIB REC Error");
                                        gpib_fault=TRUE;
                                }
                                data[19]=0;     /*** add string terminator at char 19 ***/
                                interval=atof(data+1);  /*** convert character two through 19 ***/
                                /*** send period request ***/
                                ibwrt (hp,"FN4 WA1",7);
                                if (ibsta & ERR){
                                        strcpy(sys_message,"GPIB SEND Error");
                                        gpib_fault=TRUE;
                                }
                                wait_count = 0;
                                /*** compute angle ***/
                                uut_angle =((sys.drum_gain*interval/period) + sys.drum_offset);
                                timed_avg = (timed_avg + uut_angle)-timed_avg/3;
                                uut_angle = timed_avg/3;
                                uut_linearize(uut_angle);
                                /*** restart at case 1 & flag update complete ***/
                                uut_flowmod=1;
                                update_disp1=TRUE;
                                break;
                        default:
                                uut_flowmod = 0;
                                break;
                        } /*** endswitch uut_flowmod ***/
                break; /*** timed ***/
        default:        /*** indicator type ***/
                strcpy (sys_message,"Wrong UUT TYPE");
                break;
        } /*** endswitch ind type ***/
}


/***************************************
                UUT LINEARIZE
        Used by uut_flowcalc.  Calculates flow based
```

```
        on angle-to-flow table data stored for each UUT.
        Requires lin_calc() to have been previously
        executed to set up gain table between breakpoints.
***************************************/
uut_linearize(double degrees){
        int i;
        float delta_deg,temp;
        static int last_point = 0;

        if (i = last_point){
                for (  ;i>-1;i--){
                        if(uut[table_num].angle[i] < degrees)
                        break;
        }       }
        for (++i;i<data_length;i++){
                if((uut[table_num].angle[i]-degrees) >0)
                break;
        }
        last_point =(--i);
        delta_deg = degrees - uut[table_num].angle[i];
        uut_flow = uut[table_num].pph[i] + (delta_deg*uut_gain[i]);
        flow_diff = uut_flow-sys_flow;
}

/*********************************************
                MEMORY COPY ROUTINES
        Copies to and from temporary memory during
        calibrations so previous values are not corrupted
        Used by program_mode & cal_flows
*********************************************/
copyto_temp(void){
        int i;
        memcpy(&uut_temp,&uut[table_num],sizeof(struct uut_data));
        memcpy(&sys_temp,&sys,sizeof(struct sys_table));
        memcpy(&temp_calup,&cal_up,sizeof(cal_up));
        memcpy(&temp_caldown,&cal_down,sizeof(cal_down));
        memcpy(&temp_calback,&cal_back,sizeof(cal_back));
#ifdef DEBUG
         for (i=0;i<17;i++)
                        printf("%d  %f\n",i,1000000*cal_back[i]);
#endif
```

```
}

copyto_mem(void){
        int i;
        memcpy(&uut[table_num],&uut_temp,sizeof(struct uut_data));
        memcpy(&sys,&sys_temp,sizeof(struct sys_table));
        memcpy(&cal_up,&temp_calup,sizeof(cal_up));
        memcpy(&cal_down,&temp_caldown,sizeof(cal_down));
        memcpy(&cal_back,&temp_calback,sizeof(cal_back));

#ifdef DEBUG
         for (i=0;i<17;i++)
                        printf("%d  %f\n",i,100000*cal_back[i]);
#endif

}

/*****************************************************

                ADJUST_FLOW
        Adjusts flow to specified pph.  Calculates size of
        valve required.  Tries to use previous valve size.
*****************************************************/
adjust_flow(double pph){
        int dp,v;

        if (pph>120000) pph = 120000;
        if (pph<0) pph=0;
        if ((pph>SMALL_MAX) && (mastermeter==LOW))
                master_large();
        flow_value = pph;               /*** copy to global variable ***/
        if (pph < SYS_FLOW_MIN)
                setvalve(0,0);
        else if (pph > last_pph){               /*** increasing flow ***/
                memcpy(&calpoint,&cal_up,sizeof(calpoint));
                switch(last_valve){
                        case(0):
                        case(1):
                                if (pph <= cal_up[0][16]){
                                        setvalve(1,pph);
```

```
                                break;
                        }
                case(2):
                        if (pph <= cal_up[1][16]){
                                setvalve(2,pph);
                                break;
                        }
                case(3):
                        setvalve(3,pph);
        }       }
        else{                                   /*** decreasing flow ***/
                memcpy(&calpoint,&cal_down,sizeof(calpoint));
                switch(last_valve){
                case(3):        /** large valve bad at low flow so use medium **/
                        if (pph >= cal_down[1][16]-1000){
                                setvalve(3,pph);
                                break;
                        }
                        memcpy(&calpoint,&cal_up,sizeof(calpoint));
                case(2):
                        if (pph >= cal_down[1][0]){
                                setvalve(2,pph);
                                break;
                        }
                        memcpy(&calpoint,&cal_up,sizeof(calpoint));
                case(1):
                case(0):
                        setvalve(1,pph);
        }       }
        last_pph = pph;         /*** save for next time ***/
        return(1);
}

/******************************************************
                SETVALVE
        Sets specified valve to specified pph.
        Safely sequences to new flow control valve.  Closes
        all valves before opening a new valve.  Calculates
        milliamps required to get specified pph.  Uses 17 point
        linearization curve (from "calibration") for each valve.
*******************************************************/
```

```
setvalve(int valve_num,float flow){
        int i,dp,press_turnoff=FALSE;
        float current;

        if((flow > SMALL_MAX) && (mastermeter == LOW)){
                disp(2,"Flow Too HIGH!!");
                delay (2000);   /*** lock up every thing for 2 seconds ***/
                flow = SMALL_MAX;
        }
        if (valve_num == 0){
                vclose(1);
                vclose(2);
                vclose(3);
                ma_out(valve_io[0],0);  /*** set all valves to minimum ***/
                ma_out(valve_io[1],0);
                ma_out(valve_io[2],0);
        }
        if (valve_num != last_valve){
                io_output(BYPASS_VALVE,ON);     /*** bypass turnon surge around UUT ***/
                disp(2,"CHANGING VALVES");
                vclose(1);      /*** if different flow valve shut all off first ***/
                vclose(2);
                vclose(3);
                ma_out(valve_io[0],0);  /*** set all valves to minimum ***/
                ma_out(valve_io[1],0);
                ma_out(valve_io[2],0);
                if(valve_num >= 2){             /*** if valve 2 or 3 ***/
                        ma_out(BKPRESS_OUT,0);  /*** turnoff backpressure during valve change ***/
                        delay (1000);
                        press_turnoff = TRUE;
                }
        }
        if (valve_num--){   /*** note dec valve number to match array ***/
                current = 0;  /*** default if table out of range ***/
                for (dp=0;dp<16;dp++){
                        if (flow_value < calpoint[valve_num][dp+1]){
                                current = dp + ((flow_value -calpoint[valve_num][dp])
                                                /(calpoint[valve_num][dp+1]-calpoint[valve_num][dp]));
                                break;
                }       }
                ma_out(valve_io[valve_num],current);
```

```
                vopen (valve_num +1);
        }
        if (press_turnoff){
                delay(1000);
                ma_out(BKPRESS_OUT,back_ma);     /*** reset previous backpressure current ***/
        }
        if (++valve_num != last_valve){ /*** if same as last time ***/
                delay(1500);  /**msec for turnon surge to pass ***/
                disp(2,"READY");
        }
        if (!bypass_on) turn_bypass(OFF);
        last_valve = valve_num;
}

/********** OPEN & CLOSE VALVES **********/
vopen(int valve_num){
        switch (valve_num){
                case (1):
                        io_output(SMALL_VALVE,OPEN);
                        break;
                case (2):
                        io_output(MEDIUM_VALVE,OPEN);
                        break;
                case (3):
                        io_output(LARGE_VALVE,OPEN);
                        break;
        }
}

vclose(int valve_num){
        switch (valve_num){
                case (1):
                        io_output(SMALL_VALVE,CLOSE);
                        break;
                case (2):
                        io_output(MEDIUM_VALVE,CLOSE);
                        break;
                case (3):
                        io_output(LARGE_VALVE,CLOSE);
                        break;
        }
```

```
}

/*********************************************

                EXTERNAL CONTROL
        Reads Remote Input jack voltage (front panel),
        scales the value and outputs to presently selected
        regulator valve.  Limits flow to MAX UUT value.
*********************************************/
external_control(void){
        float flow_val;
        flow_val = (analog_in(EXTRNCHAN)*uut[table_num].max_flow*sys.external_gain);
        if (flow_val > uut[table_num].max_flow) /*** check just to be sure ***/
                flow_val = uut[table_num].max_flow;
        adjust_flow(flow_val);
#ifdef DEBUG
        printf("                           External flow = %f",flow_val);
#endif
        serial_in(DISP2PORT);           /**** flush display buffers ***/
        serial_in(DISP1PORT);
}


/*****************************************************
                CALIBRATE FLOW REGULATORS
        Sequences entire range of current control outputs for
        each of three flow regulating valves.  Sequences from
        max to min, then min to max.  Then calculates flow required
        to obtain approx 7 psi backpressure for the full range
        of backpressure control current outputs.
*****************************************************/

cal_flow(void){
        int v,dp,i,j,k;
        float lastpress,newflow,nextflow,deltaflow,temp_bp;
        float bp_save[4][17];

        setvalve(0,0);                  /*** initialize all valves off ***/
        ma_out(BKPRESS_OUT,back_ma = 0);
        disp(2,"TURNON Pump");
        while(pump_on ==FALSE){
```

```
        switch_input();
        alarm_proc();
        if (prog_mode ==OFF)
                return(FAULT);
}
turn_bypass(OFF);  /*** insure no bypass ***/

/*** Calibrate from max to min flow ***/
master_large();
temp_bp = 4;
for (v=3;v>0;v--){                       /*** begin cal of large ***/
        temp_bp -= 4;   /*** back off for next valve max ***/
        ma_out(BKPRESS_OUT,back_ma = temp_bp);
        setvalve(v,0);
        for (dp=16;dp>=0;dp--){         /*** max flow first ***/
                sprintf(msg2,"Valve% d step %d",v,dp+1);
                disp(2,msg2);
                ma_out(valve_io[v-1],(float)dp);
                for (k=0;k>40;k++){
                        if (cal_delay(1500) == FAULT)
                                return(FAULT);
                        if (backpressure(20) > 5)
                                break;           /*** enough backpressure ***/
                        temp_bp += 0.1;         /*** else bump up slightly ***/
                        ma_out(BKPRESS_OUT,back_ma = temp_bp);
                }
                bp_save[v][dp]= temp_bp;
                if (cal_delay(12000) == FAULT)
                        return(FAULT);
    if (dp==16){                               /*** extra time for first entry ***/
                if (cal_delay(5000) == FAULT)
                        return(FAULT);
    }
                temp_caldown[v-1][dp] = sys_flow;
                if(sys_flow < 15000){
                        if (mastermeter == HIGH){
                                master_small();
                                if (cal_delay(8000)==FAULT)
                                        return(FAULT);
                        }
                }
```

```
        }
        setvalve(0,0);          /*** force all off for safety ***/
}

/*** Calibrate from min to max flows ***/
master_small();
for (v=1;v<4;v++){                     /*** begin cal of small ***/
        ma_out(BKPRESS_OUT,back_ma = bp_save[v][0]);
        setvalve(v,0);
        for (dp=0;dp<17;dp++){          /*** min flow first ***/
                sprintf(msg2,"Valve %d step %d",v,dp+1);
                disp(2,msg2);
                ma_out(BKPRESS_OUT,back_ma = bp_save[v][dp]);
                if (cal_delay(2000) == FAULT)
                        return(FAULT);
                ma_out(valve_io[v-1],(float)dp);
                if (cal_delay(12000) == FAULT)
                        return(FAULT);
    if (dp==0){                         /*** extra time for first entry ***/
                if (cal_delay(7000) == FAULT)
                        return(FAULT);
    }
                temp_calup[v-1][dp] = sys_flow;
                if(sys_flow > 10000){
                        if (mastermeter == LOW){
                                master_large();
                                if (cal_delay(8000)==FAULT)
                                        return(FAULT);
                        }
                }
        }
        setvalve(0,0);          /*** force all off for safety ***/
}
/***** Store new flow control values for use by backpressure cal *****/
copyto_mem();

/***** Calibrate Backpressure Valves *****/
ma_out(BKPRESS_OUT,back_ma = 16);               /*** start valve ***/
adjust_flow(newflow = 100);
if(cal_delay(10000)==FAULT)
        return(FAULT);
```

```
master_small();
for (i=16;i>=0;i--){
	ma_out(BKPRESS_OUT,back_ma = (float)i);
	adjust_flow(newflow=newflow*1.2);
	if(cal_delay(8000)==FAULT)
		return(FAULT);
	for (j=0;j<4;j++){
		nextflow = newflow*sqrt(7.0/backpressure(20));
		deltaflow = nextflow-newflow;
		newflow = nextflow -deltaflow/3;
		if(newflow > 20000){
			if (mastermeter == LOW){
				master_large();
				if (cal_delay(8000)==FAULT)
					return(FAULT);
			}
		}
		adjust_flow(newflow);
		if (cal_delay(8000)==FAULT)
			return(FAULT);
		sprintf(msg2,"BP%d Flo%6.0f",i,sys_flow);
		disp(2,msg2);
	}
	sprintf(msg2,"BP%d FinF%6.0f",i,sys_flow);
	disp(2,msg2);
	cal_delay(5000);	/*** extra delay before final data ***/
	temp_calback[i]=(backpressure(30)-sys.bp_min_press)/(pow(flow_value,sys.bp_exponent));
}
adjust_flow(0);
/***** Store new backpressure control values  *****/
copyto_mem();

/******* Calibration complete *******/
master_large();		/*** force large for safety ***/
turn_bypass(ON);	/*** reset to safe position ***/
disp(2,"TURNOFF Pump");
while(pump_on){
	switch_input();
	alarm_proc();
}
return(NOFAULT);
```

```
}


/*****************************************
                CALIBRATION DELAY
        Used by cal_flow to delay the specified
        number of milliseconds.  Constantly check
        switch inputs alarms and updates the master
        flow meter.
* *****************************************/

cal_delay(int msec){
        int i,j;

        for(j=1;j<msec/5;j++){
                delay(4);       /*** msec ***/
                switch_input();
                alarm_proc();
                if (system_error | !pump_on){
                        master_large();         /*** force large for safety ***/
                        setvalve(0,0);          /*** force all off for safety ***/
                        turn_bypass(ON);        /*** reset to safe position ***/
                        ma_out(BKPRESS_OUT,back_ma = 0); /*** remove backpressure ***/
                        return(FAULT);
                }
                master_flow(); /*** update each loop ***/
        }
        return(NOFAULT);
}


/********************************************
                BACKPRESSURE SERVICE
        PID process uses constants entered during program
        mode.  Sets Integral to desired pressure current on
        the first (and only first) time entered.
********************************************/
int bp_service(int init_cmd){
        static long int last_time;
        int temp;
        float error,update;
```

```
	static float last_error;

	if ((flow_value < 40) || !pump_on || bypass_on){  /*** if no flow ***/
		return(0);
	}
	if (init_cmd == BP_INIT){
		bp_integral = bp_val/sys.bp_I;
		bp_set(bp_val);
		last_error = 0;
		last_time = ttime;	/*** force update with new values ***/
	}
	else{
		if (system_mode != MANUAL)
			return(0);
		if ((temp = (int)(ttime-last_time))< 0)
			/***continue***/;
		else{
			if(temp<pid_period)	/*** pid_period calculated during initialization ***/
				return(0);
		}
	}
	update = sys.bp_P*(error = bp_val - backpressure(10))
			+ sys.bp_I*(bp_integral += sys.bp_Time*error)
			+ sys.bp_D*(error-last_error);
#ifdef DEBUG
	printf("err=%f  int=%f  dif=%f >%f  ",error,bp_integral,error-last_error,update);
#endif
	last_error = error;
	last_time = ttime;
	if (init_cmd !=BP_INIT)
		bp_set(update); /*** output new ma for adjusted pressure ***/
}

/*********************************************
		BACKPRESSURE SET
	Calculates backpressure current required to
	achieve specified backpressure at the present
	flow value.  Uses 16 point table of backpressure
	cal values.
*********************************************/
void bp_set(float pressure){
```

```
	int i;
	float restriction,current=16; /*** initialize to max current ***/

	if(((pressure > 0.2) && (pressure < sys.bp_min_press)))
		pressure = sys.bp_min_press + 0.2;	/*** must have something ***/
	restriction = (pressure-sys.bp_min_press)/(pow(flow_value +1,sys.bp_exponent));
	for (i=1;i<16;i++){		/*** note skip 0 ma ***/
		if (restriction < cal_back[i+1]){
			if(cal_back[i+1]-cal_back[i]){  /*** test for divide by 0 ***/
				current = (float)i + ((restriction-cal_back[i])
					/((cal_back[i+1]-cal_back[i])+1e-10));
			}
			else
				current=0;
			break;
		}
	}
	ma_out(BKPRESS_OUT,back_ma = current);  /*** save current for adjust_flow ***/
}

/***********************************************
		BACKPRESSURE read & average
	Averages specified number of backpressure readings
	to simulate a low pass filter.
************************************************/
float backpressure(int avgs){
	int i;
	float pressure=0;

	if (avgs < 1)
		avgs = 1;
	for (i=0;i<avgs;i++)
		pressure += analog_in(BKPRESS_IN);
	bkpress = sys.bp_disp_gain*pressure/(float)avgs; /*** store & return value ***/
	if (bkpress>0.1)
		return(bkpress);
	else
		return(0.1);
}

/*********************************************
```

```
                ALARM PROCESSING
        Module used by main and calibration modules.
        Requires switch_input() to read alarm switches.
        Flashes horn and light once every 1200 entries,
        if fault exists.  Generates a sys_message if
        external switches open or if GPIB or RS232
        fault.  Does not remove message when fault clears.
        Waits for 10 sequential fault inputs before
        indicating a fault, to filter out noise.
**********************************************/


int alarm_proc(void){
        static int last_alarm_bits, working_alarm_bits,
                        seq_fault_count=0,horn_count;

        /********* Flash & Beep alarm if flags set *********/
        if (++horn_count > 1200){               /*** all off once a second ****/
                horn_count = 0;
                io_output(ALARM_LIGHT,OFF);
                io_output(ALARM_HORN,OFF);
        }
        if (alarm || horn){                             /*** if flag, turn on at 1/2 second ***/
                if (horn_count == 600){
                        if (horn)  io_output(ALARM_HORN,ON);
                        if (alarm) io_output(ALARM_LIGHT,ON);
                }
        }
        /******* test for new alarm condition ********/
        alarm_bits = alarm_bits & ALARM_MASK;  /*** mask for only alarm inputs **/
        if (gpib_fault)
                alarm_bits = alarm_bits | GPIB_FAULT_BIT; /*** add gpib fault bit ***/
        if (alarm_bits == working_alarm_bits)
                                return(0);
        if (alarm_bits == last_alarm_bits){
                if (++seq_fault_count > 10){
                        seq_fault_count = 0;
                        working_alarm_bits = alarm_bits;
                        if (!alarm_bits){
                                horn = alarm = OFF;
                        }
```

```
                        else{
                                if (alarm_bits & GPIB_FAULT_BIT);  /* sys_message = previous gpib message */
                                else if (alarm_bits & WATER_FAULT)  strcpy(sys_message,"WATER PRESSR LOW");
                                else if (alarm_bits & AIR_FAULT)  strcpy(sys_message,"AIR FLOW LOW");
                                else if (alarm_bits & TEMP_FAULT)  strcpy(sys_message,"OVERTEMP FAULT");
                                else if (alarm_bits & LEVEL_FAULT)  strcpy(sys_message,"OIL LEVEL LOW ");
                                else if (alarm_bits & VENT_FAULT)  strcpy(sys_message,"VENT FLOW LOW");
                                horn = alarm = ON;
                        }
#ifdef DEBUG
        printf("%s\n",sys_message);
#endif
                }
        }
        else    last_alarm_bits = alarm_bits;
}


/****************************************
                DISPLAY MESSAGE
        Displays to specified terminal, the
        specified message.
****************************************/
disp(int port,char *msg){
        int i,chan=DISP1PORT;
        char ch;
        if (port == 2){
                chan=DISP2PORT;
        }
        if (msg[0] != '\x1B'){
                if (port == 2)
                        strcpy(last_msg2,msg);  /*** save for adjust flow ***/
                ch='\n';                /*** send cr ***/
                if (serial_out(chan,ch))
                        system_error = SERIAL_ERROR;
        }
        for (i=0;i<18;i++){
                if ((ch=msg[i]) == NULL)
                        break;
                if (serial_out(chan,ch))
                        system_error = SERIAL_ERROR;
```

```
        }
        ch=8;            /*** add backspace ***/
        if (serial_out(chan,ch))
                system_error = SERIAL_ERROR;
}

/*******************************************
                DISPLAY TERMINAL LIGHT DRIVERS
        Displays specified function light on terminals
*******************************************/
#define LIGHTONCMD "\x1Bh01\002\000"
#define LIGHTOFFCMD "\x1Bh00\002\000"
#define CONVRTO_LT ('A'-'1')    /*** 16 ***/

light1_on(int number){
        char string[20];

        strcpy(string,LIGHTONCMD);
        number -= CONVRTO_LT;
        string[2]=(char)number;
        disp(1,string);
}

light2_on(int number){
        char string[20];

        strcpy(string,LIGHTONCMD);
        number -= CONVRTO_LT;
        string[2]=number;
        disp(2,string);
}
void light1_off(void){
        disp(1,LIGHTOFFCMD);
}
void light2_off(void){
        disp(2,LIGHTOFFCMD);
}


/****************************************
                 RESET DISPLAY
```

```
        Resets terminals to standard conditions
*****************************************/
reset_disp1(void){
        d1mod=0;                                        /*** restart module flag ***/
        while (serial_in(DISP1PORT));   /*** clear input buffer ***/
        light1_off();
        light1_on (control_mode);
        light1_on (disp1_mode);
}

reset_disp2(void){
        while (serial_in(DISP2PORT));
        keyin2[0]=NULL;                  /*** null first char ***/
        d2mod=0;
        light2_off();
        light2_on (disp2_mode);
}


/*****************************************
                PARALLEL I/O
        Uses OPTO22 AC5 24 Channel IO card.

*******************************************/
#define IOBASE 0x220
#define PORT1_IO  IOBASE
#define PORT1_CTL IOBASE + 1
#define PORT2_IO  IOBASE + 2
#define PORT2_CTL IOBASE + 3
#define PORT3_IO  IOBASE + 4
#define PORT3_CTL IOBASE + 5

/******** PARALLEL IO INITIALIZE ********/
int parallel_io_init(void){
        output_image = 0;
        outportb(PORT1_CTL,0x00);       /*** set port 1 as all outputs ***/
        outportb(PORT1_IO,output_image); /*** imediatelly force all off ***/
        if (inportb(PORT1_CTL) != 0x80)
                system_error = system_error;    /*** ??? ***/
        outportb(PORT2_CTL,0xff);       /*** set port 2 as all inputs ***/
        if (inportb(PORT2_CTL) != 0x3f)
```

```
                system_error = system_error;
        outportb(PORT3_CTL,0xff);       /*** set port 3 as all inputs ***/
        if (inportb(PORT3_CTL) != 0x3f)
                system_error = system_error;
}

/********* PARALLEL IO OUTPUT **********/
io_output(int bitmask,int level){

#ifdef DEBUG
        printf("IO Output bit #: %d   value: %d\n",bitmask,level);
#endif

        if (level != ON)
                output_image = output_image & ~bitmask;
        else output_image = output_image | bitmask;
        outportb(PORT1_IO,output_image);
}

/***** SWITCH INPUT (Parallel I/O Inputs) ****/
void switch_input(void){
        int port2_input,port3_input;

        alarm_bits = port2_input = inportb(PORT2_IO);
        if (~port2_input & PUMP_ON_BIT  )
                  pump_on =TRUE;
         else  pump_on =FALSE;
        port3_input =~inportb(PORT3_IO);
        if (port3_input & PROGRAM_BIT)
                 prog_mode =TRUE;
         else  prog_mode =FALSE;
        if (port3_input & STOP_BIT){
                turn_bypass(ON);
        }
        if (port3_input & START_BIT){
                turn_bypass(OFF);
        }
        system_mode =port3_input & SYSTEM_MASK;
}

/***************************************
```

```
        MASTER FLOW METER SELECT
***************************************/
void master_large(void){
        io_output(MASTER_VALVE,HIGH);
        io_output(HI_LO_RELAY,HIGH);
        mastermeter = HIGH;
}
void master_small(void){
        io_output(MASTER_VALVE,LOW);
        io_output(HI_LO_RELAY,LOW);
        mastermeter = LOW;
}

/***************************************
        TURN ON/OFF BYPASS VALVE
***************************************/
void turn_bypass(int on_off){
        io_output(BYPASS_VALVE,on_off);
        bypass_on = on_off;
        if(on_off == ON)
                ma_out(BKPRESS_OUT,back_ma = 0);
}

/****************************************
                RESPONSE TEST
        Module used by main to control UUT response
        tests.  Only one step is executed at a time so
        that program will return quickly for other tasks.
        Energizing/deenergizing of bypass valve sequences
        to next step.
****************************************/
response_service(void){
        static long int start_time;
        static float resp_value;

        serial_in(DISP2PORT);           /**** flush display2 buffer ***/
        delay(100);     /*** to much data for display 2 ***/
        switch(resp_mod){
                case(0):
                        resp_value=uut_flow * 0.99;
                        resp_mod++;
```

```
                        break;
                case(1):
                        if (!bypass_on) /*** wait until flow bypassed ***/
                                break;
                        resp_mod++;
                        start_time = ttime;
                        break;
                case(2):
                        if (!bypass_on){
                                resp_mod++;
                                start_time = ttime;
                        break;
                        }
                        sprintf(msg2,"WaitTime %4.1fsec",(ttime - start_time)/18.2);
                        disp(2,msg2);
                        break;
                case(3):
                        sprintf(msg2,"RespTime %4.1fsec",(ttime - start_time)/18.2);
                        disp(2,msg2);
                        if (uut_flow > resp_value || bypass_on){
                                resp_mod=0;
                                while (serial_in(MASTERPORT));  /*** empty master buffer ***/
                        }
        } /*** endswitch ***/
}

/*********************************************
                MASTER FLOW METER UPDATE
        Module used by main & calibrate to read & decipher RS232 input
        string.  If input character, then store in string until Linefeed
        If Linefeed, process string "..SV+tttt.tt ffffff.ff gg.gggg tttt.tt<^"
          where  t = temp; f = flow; g = gravity; < = LF; ^ = CR
        Return 1 (to indicate update) if string processed, 0 otherwise
**********************************************/
#define MAX_LENGTH 42           /*** several more than 38 ***/
#define FLOW_END 21                     /*** char positions ***/
#define FLOW_START 14

int master_flow(void){

#define NUM_OF_AVG 6
```

```
        char ch;
        int i;
        static float flow_mem[NUM_OF_AVG + 1];
        float flow_avg;
        static int update_limit_ctr=0;

        /*** if no update within 10 seconds, shutdown ***/

        if(++update_limit_ctr>10000){
                strcpy(sys_message,"MASTER FLOW Err");
                gpib_fault=TRUE;
        }
        while (1){
                if (ch = serial_in(MASTERPORT)){
                        if ((master_msg[++msg_pointer] =ch) ==LF){
                                msg_pointer = 0;
                                if ((master_msg[3] != 'S') || (master_msg[4] != 'M'))
                                        return(0);
                                master_msg[FLOW_END]='\0';
                                sys_flow=atof(master_msg+FLOW_START);

                                flow_avg = 0;
                                for (i=NUM_OF_AVG - 1;i>=0;i--){
                                        flow_avg += flow_mem[i];
                                        flow_mem[i+1] = flow_mem[i];
                                }
                                flow_mem[0] = sys_flow;
                                sys_flow = flow_avg/NUM_OF_AVG;
                                master_update=TRUE;
                                update_limit_ctr=0;
                                return(1);
                        }
                        if (msg_pointer >= MAX_LENGTH)
                                msg_pointer = 0;
                }
                else
                        return(0);
        } /*** end while ***/
        return(0);
}
```

```
/*********************************************
              SERIAL INITIALIZE
        Initialize all 4 serial channels
*********************************************/
void serial_init(void){
        int status;
         /****  9600  8bits  1stop  noparity ****/
        if (bioscom(0, (0xE0 | 0x03 | 0x00 | 0x00)    ,0) & 0x8000)
                system_error = SERIAL_OPEN_ERROR;
        if (bioscom(0, (0xE0 | 0x03 | 0x00 | 0x00)    ,1) & 0x8000)
                system_error = SERIAL_OPEN_ERROR;
        if (bioscom(0, (0xE0 | 0x03 | 0x00 | 0x00)    ,2) & 0x8000)
                system_error = SERIAL_OPEN_ERROR;
        if (bioscom(0, (0xE0 | 0x03 | 0x00 | 0x00)    ,3) & 0x8000)
                system_error = SERIAL_OPEN_ERROR;

}

/*********************************************
              SERIAL INPUT SINGLE CHARACTER
*********************************************/
int serial_in(int chan){
        if (bioscom(3,0,chan) & 0x100)    /*** if DATA READY ***/
                return(bioscom(2,0,chan));
        return(0);
}

/*********************************************
              SERIAL OUTPUT SINGLE CHARACTER
        Return with non-0 if not transmitted
*********************************************/
int serial_out(int chan,int data){
        int status;
        status = bioscom(1,data,chan);
        return (status & 0x8000);    /*** Mask for TIMEOUT bit ***/
}


/*********************************************
              ANALOG INITILIZE
```

```
        Initialize Action Pak AICP analog satalite
        through RS232 commands & industry standard language
*********************************************/
void analog_init(void){
        char responce[20];

    /*** power up clear, expansion area ***/
    action_pak(">04A",responce);

    /*** power up clear, digital area ***/
    action_pak(">44A",responce);

    /*** power up clear, analog area ***/
    action_pak(">84A",responce);

        /*** configure modules as inputs for pos 14,13,12,11,10,8,6
                using 0111,1101,0100,0000.  Reduce 16 bits to the four
                hex chars "7D40", preceded by ">84H" ***/
        action_pak(">84H7D40",responce);

    /*** configure modules as outputs for pos 3,2,1,0
                using 0000,0000,0000,1111.  Rreduce 16 bits to the four
                hex chars "000F" preceded by ">84I" ***/
        action_pak(">84I000F",responce);

}

/*********************************************
                VOLTS IN
        Read single measurement of specified input channel.
        Returns value from 0 to 5.00, equivalent to 0-5 volts.
*********************************************/
float analog_in(int channel){
        char *ptr,*dest,message[20]=">84L",buffer[20],responce[20];
        int i,counts;
        float value;

    /* format position string, need 4 hex values */
    channel = 0x0001 << channel;
    itoh(channel,4,buffer);
    strcat(message,buffer);
```

```
    action_pak(message,responce);

    /* pull value out of responce */
    ptr = responce;
    dest = buffer;
    ++ptr;      /* move past first char */
    for(i = 0;i <= 3;i++,ptr++,dest++) *dest = *ptr;

    /* convert value from hex to BCD */
    for(ptr = buffer,i = 0;i <= 3;i++,ptr++){
        if(*ptr >= 'A' && *ptr <= 'F')
            *ptr = *ptr - 'A' + 10;
        else
            *ptr = *ptr - '0';
    }

    /* convert from BCD to int */
    ptr = buffer;
    counts = 0;
    counts = counts + (int) *ptr * 0x1000;
    ++ptr;
    counts = counts + (int) *ptr * 0x100;
    ++ptr;
    counts = counts + (int) *ptr * 0x10;
    ++ptr;
    counts = counts + *ptr - 0x1000;

    /* convert responce to float */
    value = (float) counts/819;
    return(value);
}

/*********************************************
                MA OUT
        Outputs to specified channel specified current,
        offset by +4 ma to achieve 4-20 ma.  Range of
        commanded current is 0-16.00 ma.
*********************************************/
void ma_out(unsigned int channel,float value){
```

```
        char *ptr,message[20]=">84J",buffer[20],responce[20];
        int counts;

        if (value<0)
                value = 0.0;
        if (value > 16.0)
                value = 16.0;
#ifdef DEBUG
        printf("ma_out %x  %f\n",channel,value);
#endif

    /* format position string, need 4 hex values */
    channel = 0x0001 << channel;
    itoh(channel,4,buffer);
    strcat(message,buffer);

    /* format data string */
    if(value > 15.999)  value = 15.999;
        counts = (int)( value * 255);
        itoh(counts,3,buffer);
    strcat(message,buffer);

    action_pak(message,responce);
}

/********************************************
                ACTION PAK
        Used by analog_init, volts_in & ma_out to
        send and receive messages from analog io hardware
********************************************/
#define COM4 3  /*** define hardware physical port ***/

void action_pak(char *msg, char *responce){
        char *ptr,*ptr2,buffer[20],message[20]="";
        int i,check_sum;

        strcat(message,msg);    /*** copy message to defined area ***/
    /*** calculate check sum and append to message ***/
    check_sum = 0;
    ptr = message;
    ++ptr;      /* start check sum after < */
```

```
        for( ;*ptr != NULL;ptr++) check_sum = check_sum + *ptr;
    check_sum = check_sum & 0x0ff;
    itoh(check_sum,2,buffer);
        strcat(message,buffer);
        strcat(message,"\r");

    /* send out message */
        ptr = message;
        while(*ptr){
                serial_out(COM4,*ptr);
        ++ptr;
        }
    /* get responce untill <cr> */
         ptr2 = responce;
         for (i=0;i<10000;i++){
                 if (*ptr2 = (serial_in(COM4) & 0x7f)){
                         if(*ptr2 == '\r') goto done;
                         ++ptr2;
                 }
         }
         strcpy(sys_message,"RS232 TIMEOUT!");
         gpib_fault= TRUE;

        done: *ptr2 = NULL;

    /* check for errors */
        if(responce[0] == 'N')  system_error = ACTION_ERROR + (responce[2]-'0');
        delay(4);       /*** wait for action pak to process message before next message ***/
}

/*********************************************
                INTEGER TO HEX
        Converts integer value to ascii hex, with specified
        "places" accuracy.  If "places" is zero
        only number needed is returned.
*********************************************/
void itoh(unsigned int value, int places, char *buff){
        static char hex[16] = {"0123456789ABCDEF"};
        char *ptr;
        int digit,length,i;
        unsigned int remainder;
```

```
    ptr = buff;
    /* fills buff in reverse order */
    do{
        digit = value % 16;
        *ptr = hex[digit];
        value = value/16;
        ++ptr;
    }while(value);
        *ptr = NULL;

    /* pad with zeros if required */
    if(places){
        length = places - strlen(buff);
        if(length){
            for(i = 0;i < length;i++,ptr++)
                *ptr = '0';

                        *ptr = NULL;
        }
    }
    strrev(buff);
}

/********************** END OF FLOW CONTROL PROGRAM **************************/
```

Industrial Computer Source

Product Manual

Model MID
Reference Manual

Industrial Computer Source 4837 Mercury St. San Diego, CA 92111 619-279-0084

Industrial Computer Source (UK) LTD P.O. Box 81 Chichester, West Sussex, PO20 7SD England 0243-533900

Industrial Computer Source France, ZA Courtaboeuf 18 Avenue du Québec BP 712 91961 LES ULIS Cedex(1)69.07.28.22

FOREWORD

This manual explains how to use the Industrial Computer Source computer which is specifically designed to be compatible with the IBM PC, XT or AT computer. The chassis is engineered for data acquisition and control applications in the rugged environment of the industrial world with built-in reliability and serviceability.

ICS supports its products with complete and thorough documentation. Technical assistance is available at (619) 279-0084.

Manual Errors, Omissions and Bugs: A Bug Sheet is included as the last page of this manual. Please use it if you find a problem with the manual you believe should be corrected.

NOTICE

The information in this document is provided for reference only. ICS does not assume any liability arising out of the application or use of the information or products described herein. This document may contain or reference information and products protected by copyrights or patents and does not convey any license under the patent rights of ICS, nor the rights of others.

OUR STATEMENT OF SALE

These are our terms and conditions of sale in plain English. This is what you can expect of us, your vendor, and what we request of you, a valued customer.

30-DAY UNCONDITIONAL GUARANTEE

"If you are not 100% satisfied for any reason within 30 days of purchase with our merchandise, you may return it at once for a prompt refund, no questions asked!"

We want your equipment to work - we guarantee it. Unfortunately, in the Personal Computer market, there are equipment combinations which will not work because of resource conflict. We try to anticipate problems, but it is impossible to test all boards in all computers with all other manufacturer's products. Or, your application may require a different approach. Whatever the cause, if you would like to return a purchase, please call. The RA procedure is listed below.

If you don't ask for a Return Authorization Number within 30 days of purchase or do not return a purchase within 10 days of receiving an RA number, a restocking charge of not less than 20% will be deducted from your credit.

If you require additional time to test the product, call your regional manager (619-279-0084) and ask for an extension. An extension will not be granted if requested after the 30 day period after purchase. For an extension, an RA number will be assigned with a notation of the extension period. The product must eventually be returned under this RA to avoid restocking charges.

WARRANTY

All items sold by INDUSTRIAL COMPUTER SOURCE are warranted against defects in parts and workmanship for 1 year. Manufacturer's warranties apply if they are for more than 1 year. If something quits working within the INDUSTRIAL COMPUTER SOURCE warranty period, we will repair it at no cost to you. This does not apply if the item is damaged by customer misuse. Repairs are effected at the INDUSTRIAL COMPUTER SOURCE facility in San Diego, California.

DEAD-ON-ARRIVALS (DOA'S)

All INDUSTRIAL COMPUTER SOURCE products are burned-in and tested. However, if you receive something that does not work when you first plug it in, call for technical assistance and we will work with you to locate the problem. If the product is faulty and under our warranty, we will issue an RA number and immediately replace the product. If you require an overnight replacement, we will need a Purchase Order to ship you the new board. On receipt of the faulty board, we will credit your PO. Normally, no charges will apply to this PO. There are two exceptions:

A) We will charge you for the freight if you ask us to ship overnight (FED EX, UPS RED, etc.). We will normally ship a replacement at our expense via UPS BLUE (second day delivery).

B) We will test the returned board and will charge you for the repairs if it is determined that customer misuse damaged the board.

FAILURES OTHER THAN DOA

If you have an INDUSTRIAL COMPUTER SOURCE product which fails within the 1 year warranty period, INDUSTRIAL COMPUTER SOURCE will repair or replace that product at our option.

1) You must receive an RA number to return the product.
2) Allow 2 weeks for a repair. We will do everything we can to return the item as soon as possible. We do not have loaners. If you need another unit while yours is being repaired, you will have to purchase a new board. Our 30 day Unconditional Guarantee does not apply if you have a board in for repair and purchase a replacement. That is, you can't purchase a board and return it when you receive your repaired unit back.
3) We will return your repaired item at our expense via UPS. You will be charged freight if you request an overnight delivery or other carrier.

REPAIRS NOT COVERED BY WARRANTY

Any damage resulting from customer misuse will be charged at $60.00/hour for labor plus parts. Customer misuse is usually typified by more than one chip failed on the same board, burnt traces, or physical damage to the board (bent pins, etc.). I/O sections also fail a lot due to excessive voltages. Be careful when connecting a system to the outside world; ground loops and unexpected voltages are more common than you think.

REPAIRS ON ITEMS OLDER THAN 1 YEAR

All out-of-warranty repairs will be charged at $60.00/hour for labor plus parts. Please call for an RA number. The limitations of in-warranty repairs apply regarding loaners, shipment charges, etc. You will be charged for return freight.

RETURN PROCEDURE

1) To return an item for credit or repair, call our Customer Service Department (619-279-0084) and ask for a Return Authorization Number. Items returned without an RA number will be refused. We need the following information to issue an RA:

A) Company name, address, etc.
B) End user name and phone number and contact name and phone number
C) Invoice number or Purchase Order number and purchase date
D) Reason for return (helps us serve you better)

It will facilitate your repair if we have the name and phone number of the end user who can answer questions concerning the reason for the return.
2) Items must be shipped to INDUSTRIAL COMPUTER SOURCE with freight prepaid. Collect items will be refused.
3) Items returned for credit must be received by INDUSTRIAL COMPUTER SOURCE within 10 days of receiving an RA.
4) Items returned for credit must be in resellable condition. That is, any special packing material must be intact, manufacturers manuals must be in "as new" condition (not bent, dirty or obviously used) and the equipment in "as new" condition. You may keep INDUSTRIAL COMPUTER SOURCE printed manuals. For example, a ZOOM modem must be returned in the ZOOM box (in good condition) with the ZOOM manual, warranty cards, disk, etc. A replacement charge of $25 will be assessed for unusable or missing manuals or boxes. Repairs to products required because of customer modification or misuse will be charged at $60.00/hour. All we ask is for you to allow us to resell the item if you don't want it. All returned items are tested to original manufacturer's specifications.

Multiline Interrupt Driver

User's Guide
Version 2.2

Rev 2.2b 8/22/88

I. INTRODUCTION

Description

MID.EXE (hereafter referred to as MID) provides fully interrupt driven, buffered input and output for multiple RS-232 serial ports installed in an IBM PC, AT, or compatible. MID is implemented as a resident BIOS extension and/or replacement. MID emulates standard BIOS interrupt 14h support as well as providing extended support in the form of block I/O functions and handshake support.

Hardware Requirements

IBM PC, XT, AT, or compatible with one or more serial ports installed. UARTS must be INS 8250, NS16450 or compatible. Ports may reside on multiple boards but must be mapped to non-conflicting port addresses. Note that the IBM bus requires that ports on separate boards be configured to separate hardware interrupt lines (IRQs).

Software Requirements

MSDOS or PCDOS version 2 or version 3. MID will not operate properly with version 1. MID may be preempted by software with integral serial support. Some software programs are ill-behaved in that they do not restore interrupt vectors and interrupt enable registers when they terminate. Use of such software will disable MID and MID must be reset to recover from such situations. Parasoft assumes no responsibility for the poor etiquette of other manufacturers' software. MID is supplied with test programs, and sample communications subroutine libraries for several popular C compilers.

Current Specifications

Fully buffered and interrupt driven on both input and output.
Low overhead, highly efficient operation.
Supports multiple boards.
- Up to eight ports per board.
- Number of boards limited only by number of available IRQs.

Character calls and block calls may be intermixed.
- Character calls emulate standard BIOS support.
- Block reads and writes provide performance for high throughput applications.

Simultaneous bidirectional I/O on all ports.
Buffer sizes individually configurable.
Configurable handshake support.
- Quick response Xon Xoff handshake support.
- Hardware (DTR/DSR) handshake support.

Estimated *peak* communications bandwidth (one board/IRQ only):

IBM PC	75,000+ bits/sec
IBM PC-AT	150,000+ bits/sec
16 Mhz 386	450,000+ bits/sec

License Agreements

MID comes with a single user license. A flat fee, no royalty OEM license is available for software developers. Any subroutine libraries may be incorporated into your product with no royalties required.

II. CONFIGURATION OPTIONS

ADDING A MULTIPORT BOARD TO A SYSTEM WITH EXISTING COM PORTS

If you add a multiport board to a computer with existing ports, there are several configuration options available.

1. Let MID support them all. Create a configuration file for each IRQ and install MID multiple times as described in section V, MULTIPLE BOARDS AND IRQS.

2. Leave COM1 (& COM2) untouched. Create a configuration file for your multiport board and install MID once. Don't tell MCONFIG about COM1 (COM2). Your program references your multiport card as ports 0...n.

   a. If the software using COM1 (COM2) has integral drivers that directly access the hardware port (most communications software, for instance), the user service interrupt may be left set to 14h. However, any software accessing COM1 (COM2) via the BIOS or DOS will instead access the first port(s) on your multiport card.

   b. If COM1 (COM2) must function through the normal BIOS, you must set MID's user service interrupt to one of the choices other than 14H. You must also change the #define for the user service interrupt in the i/o library.

In any case, you must configure the multiport card to a unique IRQ and port address map. This may not always be possible depending upon the flexibility of the card configuration, and other accessories installed in your computer. If your computer already has two ports on the motherboard, or two ports on the *same* accessory card, it may be possible to reconfigure COM2 to interrupt on IRQ4 along with COM1, thereby freeing IRQ3 for the multiport card. On an AT type machine, IRQ5 may be free if you don't have a second parallel port. If you don't have and EGA or IRMA type board, IRQ2 may be free.

Here is a list of IRQs and their typical uses. An AT contains a second cascaded interrupt controller for a total of 16 IRQs, but neither MID nor most cards permit configuration to the second controller.

IRQ	PC/XT Function	AT Function
0	Timer	Timer
1	Keyboard	Keyboard
2	Reserved (EGA, Synchronous cards(3270, 3780), LANs)	
3	COM2	COM2 (Some Novell LAN cards)
4	COM1	COM1
5	Fixed Disk	Second Parallel Port
6	Diskette	Diskette
7	Parallel Port	Parallel Port

III. CONFIGURATION, Using MCONFIG

Prior to installation a configuration file must be created which tells MID what your hardware configuration is. You may also specify startup port configurations (baud rate, etc.). If you wish MID to support ports that do not reside on the same accessory board, or if you wish to split the ports on one board between two different IRQ's, you must create a separate configuration file for each board/IRQ. Read section IV, MULTIPLE BOARDS before proceeding with configuration. *Throughout the rest of this section, comments in italics apply to multiple board installations and may be ignored by single board users.*

MID's configuration program is invoked by typing MCONFIG at the system prompt, optionally followed by a configuration file *filespec* which may include a path (up to 30 characters). If *filespec* is included, it must be separated from "MCONFIG" by exactly one space. The square braces just mean that *filespec* is optional; do not type square braces.

A:> mconfig [*filespec*]

If *filespec* is omitted, MCONFIG defaults to MID.CF in the current directory. If the configuration file is not found, MCONFIG will ask if you wish to create a new file, or whether you wish to exit.

Configuration file not found.

A - Create a new file
B - EXIT

Press B to exit, or A to create a configuration file and proceed with configuration. If you press A, the main menu will now appear.

```
Multiline Interrupt Driver Configuration - version 2
        Copyright 1985,86 - PARASOFT
              MAIN MENU

     A  - Set number of ports
     B  - Set IRQ
     C  - Set user service interrupt
     D  - Set UART base port addresses
     E  - Set buffer sizes

     F  - Set port configurations (baud, etc.)
     G  - Set handshake options, transmit timeout

     H  - Save changes and EXIT
     I  - Ignore changes and EXIT

     Enter choice:
```

Each of these choices leads to a submenu for configuring the listed option. These submenus are self explanatory and will not be reproduced here. Here is a brief explanation of each configuration option.

A - Set number of ports

Use this option to tell MID the number of ports present on the accessory board to be supported. *Enter only the number of ports that are tied to the IRQ specified by choice 'B', not the total number of ports installed in your PC.*

B - Set IRQ

MID must have some way of knowing that a port has a character available, or that the port is ready to transmit a character. The port does this by requesting service via a hardware Interrupt ReQuest line, or IRQ. The standard IBM PC and XT have eight lines for this purpose while the PC-AT has sixteen lines available. Several of these lines are used by the operating system for such things as keyboard input, disk I/O, etc. Consult your hardware documentation for information about unused lines. In most systems IRQ3 and IRQ4 are reserved for communications use and it is recommended that you choose one of these interrupts. Check your hardware documentation for information on setting the IRQ line on your communications board. This is usually done by setting DIP switches or changing jumpers. *The design of the most accessory cards requires that each accessory board containing serial ports be set to a different IRQ line.*

C - Set User Service Interrupt

While the IRQ provides an interface between the hardware and the MID interrupt routines, the User Service Interrupt is a software interrupt which provides an interface between application software running on the PC, and MID, similar to the way that the BIOS interrupts provide services for both MS-DOS and application software.

The user service interrupt may be set to one of nine choices: interrupt 14 (hex), or interrupts 60 through 67 (hex). The latter are designated as user interrupts by MS-DOS. Interrupt 14h is used for standard BIOS serial I/O support by most systems and should also be used with MID in most cases.

If you wish to make use of MID's "link" feature to support multiple boards, you must set all boards to the same user service interrupt.

INTERRUPT 14H

This selection results in MID replacing standard BIOS serial I/O support. Use this choice if you wish to run software which performs serial I/O using standard BIOS calls or device calls (AUX, COM1, COM2). This option allows the first two defined ports (first two addresses given to MCONFIG) to be accessed using the device names AUX, COM1, and COM2. AUX and COM1 usually refer to the first defined serial port, while COM2 refers to the second port. Commands such as "COPY COM1 filename.ext" should work in this mode as should "CTTY COM2".

INTERRUPTS 60H through 67H

This selection results in MID coexisting with standard BIOS serial I/O support. Use this choice if you wish to leave standard BIOS support intact. You must modify and recompile the appropriate i/o library to operate in this mode. The test programs are compiled for INT 14h, so will not work if for these choices. If you choose this option, make sure that your choice does not conflict with any other accessory installed in your PC. Options such as local area network interfaces and synchronous serial interfaces often use one or more of these interrupts.

D - Set UART base port addresses

Each serial port is physically represented by a UART chip. Each UART must be associated with a "port address" so that MID will know where the chip is. Each port requires eight bytes of I/O address space. Consult your hardware documentation for the settings used and enter those settings. The value entered here is the "base port address" and is the lowest of the eight addresses associated with the port. The menu allows you to choose one of two standard address maps, or you may specify each port individually. *Each port must have a unique address even though they may reside on separate boards.*

E - Set buffer sizes

The size of each port's transmit and receive buffers may be individually specified. Each buffer may be set to a size from 16 bytes to 60,000 bytes. However, total buffer space per copy of MID cannot exceed 60,000 bytes. Note that actual buffer capacity is one byte less than the value you specify here.

F - Set port configuration (baud, etc.)

Each port may be individually configured for baud rate, parity, number of stop bits, and number of data bits. This is done through a system of menus which requests a value for each option. These configurations take effect when MID is installed. They may be changed after installation by software using standard BIOS configuration calls. For baud rates above 9.6k, see the note in section IV, Installing MID.

G - Set handshake options, transmit timeout

Each port may be individually configured for hardware handshake, software handshake, and transmit timeout value. This is done through a system of menus which requests a value for each option. These configurations take effect when MID is installed and may be changed after installation by software using BIOS configuration calls.

MID provides a timeout for the character transmit function (function 1). This timeout is provided for DOS compatibility. The timeout is merely a program loop and actual time values will vary with processor and clock speed. The "short" timeout is approximately two seconds on a standard PC and will prevent timeouts caused by character transmission time. The "long" timeout is in the 5 to 10 minute range and will prevent timeouts caused by taking the printer offline temporarily, etc. Note that your PC will do NOTHING but sit in an idle loop during this timeout. For multiport operation, it is best to use a zero timeout and check the timeout bit every time. This leaves your program in control should a device become not ready. Better yet, use block calls. The block transmit function always returns a zero count immediately if unable to accept characters for transmission.

H - Save Configuration and EXIT

When you are satisfied that your choices correspond with your hardware and applications software, choose this option to save this configuration to disk and exit the program.

I - Ignore Changes and EXIT

If you were just exploring, or checking the current configuration, use this option to exit without rewriting the configuration file. Do not use this option the first time you run the configuration program or no configuration file will be created.

IV. INSTALLING MID

Once you have created a configuration file, you may install the interrupt driver by typing MID at the system prompt, optionally followed by a configuration file *filespec* which may include a path (up to 30 characters). If *filespec* is included, it must be separated from "MID" by exactly one space. If *filespec* is omitted, MID defaults to MID.CF in the current directory. If you are installing multiple boards, you must invoke MID once for each board. See section IV, MULTIPLE BOARDS.

A:>mid [*filespec*]

MID will display a signon message and return the system prompt. If the configuration file is not found, MID will display a warning message and abort installation.

If MID has already been installed a reset will be performed, restoring MID to the configuration in effect when initially installed. If MID dies following execution of application software, try typing MID at the system prompt to restore MID. Any changes made to the configuration file since MID was initially installed will not be recognized.

MID requires approximately 2K of memory plus buffer space as specified to MCONFIG. If inadequate memory is available, MID will print an error message and abort installation.

Higher Baud Rates

MIDHS.EXE is the same as MID.EXE with the exception that:

19.2 kb is substituted for 110 baud (init byte 000xxxxx)
38.4 kb is substituted for 150 baud (init byte 001xxxxx)
57.6 kb is substituted for 300 baud (init byte 010xxxxx)
115.2kb is substituted for 600 baud (init byte 011xxxxx)

V. MULTIPLE BOARDS AND IRQS

Ports residing on multiple accessory boards are supported by installing multiple copies of MID, one copy for each board. Ports residing on a motherboard should be treated as a separate accessory board. In general, the procedure is to configure each board for a separate IRQ (required by the most board designs), and then use MCONFIG to create a separate configuration file for each board. MID is then invoked once for each board, specifying the corresponding configuration file on the command line when MID is invoked. MID will automatically link multiple installed copies so that the same user service interrupt may service all of the boards.

Procedure to support multiple IRQs

1. Configure each board so that all ports on a given board interrupt on the same IRQ. Choose a different IRQ for each board (See guidelines below). Consult your hardware manufacturer's documentation for information on how to set configuration jumpers or switches for your chosen IRQs.

2. Consult the hardware documentation for each board for information on available port I/O address map options. You must set the address maps for each board so that no two ports share the same base address. This may not be possible, depending upon the choices available on your boards.

3. Execute MCONFIG to create a separate configuration file for each board (IRQ). Use a unique name for each configuration file. Configure each board as if it were the only board being supported. Be sure to set the user service interrupt (normally 14h) the same for each board, and set the IRQ different for each board, corresponding with your switch settings.

4. Install MID once for each board (IRQ), specifying the corresponding configuration file on the command line.

   e.g. C:>mid board1.cf
   C:>mid board2.cf

   The first installation is a normal install. The second installation should result in a "MID link successful" message. If a copy of MID is already servicing the IRQ, a reset will be performed on that copy of MID only and no new link will be established.

   Port numbers for each board are numbered sequentially as installed, starting with zero. In the above example, if board1 is configured for 2 ports and board2 for 4 ports, then board1 ports are accessed as port 0 (COM 1) and port 1 (COM 2), and board2 ports would be accessed as ports 2 through 5.

   The order in which you install determines the port number assignments. If you reversed the two lines in the example above, board2 would be assigned as ports 0 through 3, and board1 would be assigned port numbers 4 and 5.

   Any number of copies of MID may be linked, each supporting a different IRQ and up to eight ports. The practical limit on number of ports is the number of IRQs you can scrounge.

CAUTION

Since IRQs are serviced on a priority basis, it is very easy for activity on one IRQ to swamp a lower priority IRQ causing receive data loss. Tests indicate that this begins to occur with 3 ports operating at 9600 bps on the higher priority board (5 mhz 8088). If your application requires multiline highspeed input, you should consider purchasing hardware with multiple ports on one accessory card. Products are available with up to eight ports.

Configuration Guidelines

- Configure the board with fewest ports to the higher priority (lower number) IRQ. For example, if you already have a standard COM1 and wish to add a multiport board, you might benefit by reconfiguring your COM1 to be COM2 (serviced by IRQ 3), and setting your multiport board to interrupt on IRQ 4.

- Connect devices requiring heavy transmit activity (printer, e.g.) to the lower priority (higher number) IRQ board. This prevents transmit activity from interfering with receive activity on the higher priority board.

- Connect devices generating high speed receive activity to the higher priority (lower number) IRQ board. This is sort of the inverse of the previous rule.

EXAMPLE:
IRQ 3
board with 2 ports
high speed input devices
IRQ 4
board with 6 ports
high speed output devices

- Some experimentation may be necessary to determine the most efficient configuration for your particular situation.

VI. RS-232 PRIMER

This section presents the basics of RS-232 serial communication as they relate to the IBM PC. The discussion is very basic and omits aspects of RS-232 which don't commonly arise in the PC environment.

Serial Data Transmission

The term *serial data transmission* refers to the fact that the bits composing a data byte are transmitted down a single wire one after the other. In fact, two wires are provided, one for transmission in each direction. This is in contrast to *parallel data transmission* where eight or even sixteen wires are used so that all the data bits composing the data byte may be transmitted at the same instant. Serial and parallel interfaces both contain additional wires which are used for device control, status reporting, and flow control.

Synchronization

The term *synchronization* refers to the fact that the receiving device must have some way of knowing when valid data arrives so that it can read the input signals correctly.

> **Asynchronous serial interfaces** (RS-232) achieve synchronization by the use of *framing bits*. Framing bits consist of a *start bit* which precedes the data bits, and one or two *stop bits* which follow the data bits. The receiving device synchronizes when the start bit arrives, clocks in the data bits at the appropriate time, and confirms the end of the data byte by reading the stop bit. A typical *asynchronous transmission frame* consists of ten bits composed of a start bit, followed by eight data bits (or seven data bits and a *parity bit*), followed by one stop bit.

> **Synchronous serial interfaces** of the RS-232 variety achieve synchronization through a separate clock control line. The devices must agree as to which one will control the clock line; the other device will drive its circuitry from the controlling device's clock signal. The advantage is that since both devices are driven by the same clock, more reliable and higher speed data transfer can be attained than using asynchronous transmission. Disadvantages include higher cost interfaces and more complex programming. Synchronous interfaces are common among mainframes and minicomputers where the higher performance justifies the additional cost. Synchronous interfaces are also available as options for the IBM PC and are a common way of interfacing micros to mainframes. MID does not support synchronous interfaces, nor does the hardware chip (INS 8250) commonly used for asynchronous communications with the IBM PC.

Transmission Rate (Baud or Bps)

The term *transmission rate* refers to the rate at which bits are transmitted, hence the term *bits per second*, abbreviated bps. *Baud rate* is more specifically defined as the number of signal modulations per second and may be different than bps in the case of phase shift modulated modems. In spite of this, the terms are usually used interchangeably and in fact, baud rate and bps will be identical for RS-232 transmission. The transmitting device and the receiving device must be configured for the same baud rate or else the receiving device will clock in the data bits erroneously resulting in a *framing error* or *parity error* or both. Intercharacter time is typically about equal to character transmission time and actual throughput may be lower than expected if the software driver is inefficient or if the transmitting device's CPU is burdened by higher priority tasks resulting in greater intercharacter times.

Character Error Checking (Parity)

Parity checking entails reserving one data bit for use as a *parity bit* that is set to a zero or one as necessary, so that the resulting eight bit data byte contains an even (*even parity*) number of one bits, or an odd (*odd parity*) number of one bits. Parity checking offers a crude level of error detection at best and complicates the transmission of binary (8 bit) data since only 7 bits are available for data. Again, both devices must be set to the same convention to work properly.

Frame Specification

For successful communications to take place, both devices must expect the same *frame specification*. A frame specification simply describes the number and meaning of the bits composing one transmission frame. Most devices require one start bit, either eight data bits and no parity bit or seven data bits and one parity bit (specify odd or even parity), and one stop bit. Two stop bits are sometimes used, particularly at slow (300 baud) transmission speeds.

Handshaking

If the receiving device is unable to process incoming data as fast as the transmitting device is sending, it must have some way of telling the transmitting device to suspend transmission temporarily while it catches up. When it is again ready to process incoming data, the receiving device notifies the transmitting device to resume transmission. This process is termed *handshaking* and there are two varieties.

Hardware handshaking denotes use of a separate wire in the RS-232 cable for on/off signaling from the receiving device to the transmitting device. If transmission is occurring in both directions and handshaking is desired in both directions, then two control wires must be used, one for control in each direction. MID supports DSR transmit handshaking and/or DTR receive handshaking.

Software handshaking relies on transmission of special control characters rather than separate control lines. This simplifies cabling since usually only four wires are required; transmit data, receive data, signal ground, and chassis ground. The control characters most commonly used are the XON character, also referred to as DC1, and the XOFF character, also called DC3. XON is equivalent to *control-Q* and XOFF is equivalent to *control-S*. When the receiving device wants the transmitting device to suspend transmission temporarily, it transmits an XOFF character. When it is again ready to process incoming data, the receiving device transmits an XON character to signal its readiness. Software handshaking is best suited for ASCII data as binary data may include embedded control characters. This may not be a problem if binary transmission is occurring in one direction only.

RS-232 Physical Connection

Although the RS-232 connection has traditionally been implemented using the ubiquitous 25 pin DB-25 connector, reduced reliance on hardware control signals has led to a trend toward the 9 pin DB-9 connector, most notably on the IBM PC-AT computer. Unless explicitly stated otherwise, all references to connector pin numbers refer to the conventional DB-25 connector.

Although a wider range is permitted, actual signals tend to be in these DC voltage ranges:

SIGNAL	BINARY	SIGNAL	FUNCTION	TEST LED
+ 8 to 10 volts	zero	space	on/true	red
- 8 to 10 volts	one	mark	off/false	green

The first column is actual voltage to signal ground (pin 7) as might be measured with a voltmeter. The second column shows how this signal is interpreted when measured on a data line. The *SIGNAL* column indicates alternate terms for data line signals. While idle, and between characters, data lines normally maintain negative voltage, that is, binary one, or mark. A transmission frame begins with a binary zero, that is, signal space, or positive voltage. The *FUNCTION* column shows the interpretation when the measured line is a control line rather than a data transmission line. The last column shows the color of LEDs conventionally used on test equipment to indicate the state of monitored lines.

Test Equipment

Problems in interfacing two devices which otherwise appear to be configured compatibly (baud rate, frame format, etc.) are best resolved using test equipment which is inserted inline in the cable path between the two devices. *Status Activity Monitors* (SAMs) display the state of common interface lines and are relatively inexpensive. *BreakOut Boxes* (BOBs) are more sophisticated devices that permit jumpering, or crossing of interface lines as well as providing monitor lights. The better devices will have colored LEDs to indicate the polarity of the signals, a very useful feature.

RS-232 Cabling

The RS-232 signals are not symmetrical. There is a male and female, if you will, that is not often related to the actual gender of the physical cable connector. The terms used to describe this distinction are *Data Terminal Equipment* (DTE) and *Data Communications Equipment* (DCE). These terms date from the common usage of terminals (the DTE) connected to a modem (the DCE) for communications with a remote mainframe computer. Whether a device is a DTE or DCE is usually determined by the wiring of the RS-232 connector on the device although some devices are configurable via jumper blocks, switches, or even software. Most accessory ports sold for the IBM PC are configured as DTE, most modems are DCE, printers come in both varieties.

The commonly used lines in a PC environment are:

DTE 9 pin	DTE 25 pin	Direction	DCE 25 pin	Function
	1	<------->	1	Frame ground
3	2	---->	2	Transmit data (TD)
2	3	<----	3	Receive data (RD)
7	4	---->	4	Request to send (RTS)
8	5	<----	5	Clear to send (CTS)
6	6	<----	6	Data set ready (DSR)
5	7	<------->	7	Signal ground
1	8	<----	8	Carrier detect (CD)
4	20	---->	20	Data terminal ready (DTR)
9	22	<----	22	Ring Indicator (RI)

Pin 2 (data from DTE to DCE), pin 3 (data from DCE to DTE), and pin 7 (signal ground) form the basis of the interface. If no hardware handshaking is required, a three wire or four wire cable will suffice, the frame ground connection sometimes being unavailable or unnecessary. Be sure to cross wire pins 2 and 3 when connecting devices of like type (DTE to DTE, or DCE to DCE).

The other lines are variously used to report modem status to the DTE or to indicate device ready status. Usually one line in each direction is sufficient for hardware handshake control. Several different conventions are commonly used for handshake control, but most can be made to work together by wiring a cable that presents the ready signal from one device to the appropriate pin of the other device.

MID supports DSR transmit control and DTR receive control. In practical terms, this means:

Transmit control - Connect the *ready* line from the receiving device to pin 6 (DSR) of the PC's RS-232 connector.

Receive control - Connect pin 20 (DTR) of the PC's RS-232 connector to the device ready input of the transmitting device. MID will raise or lower this line to indicate whether it has buffer space to receive additional characters.

Cable Troubleshooting

Test equipment will help diagnose difficulties caused by cable wiring. Insert the test instrument in the transmission line and observe activity of the lines. Refer to your test instrument instructions for details on how to monitor and jumper transmission lines. The following discussion assumes that the PC's port is configured DTE. Pin numbers refer to the DB-25 connector only.

The most common cable wiring problems are:

Symptom - Transmission doesn't work in either direction. BOB or SAM shows signal on one of RD or TD but no signal at all on the other.

Diagnosis - Connecting two like devices (DTE to DTE, or DCE to DCE) requires that pins 2 and 3 be crossed in the cable. A cable with pins 2 and 3 crossed is called a *null modem* cable.

Solution - Use the BOB to cross pins 2 and 3, or reverse pins 2 and 3 in one cable end and observe monitor for mark condition (green LED) on both lines.

*****

Symptom - Remote device not receiving data transmitted by PC. RD and TD both show mark condition (green LED) continuously, even when an attempt is made to transmit data from PC.

Diagnosis - Hardware handshake not satisfied. Check that pin 6 at the PC end shows true (red LED). If not, force pin 6 true by jumpering to a true (red LED) pin, pin 4 for instance, and see if this results in activity on the transmit data (TD) line.

Solution - Refer to the documentation on the remote device to determine which pin it uses for ready status and connect to pin 6 at the PC connector.

*****

Symptom - Data transmitted by remote not being received by PC. RD and TD both show mark condition (green LED) continuously.

Diagnosis - Hardware handshake not satisfied. If the remote device is DCE, make sure that pin 20 is connected straight across to pin 20. If the remote is DTE, try jumpering pin 20 (PC end of cable) to pin 5 or 6, or *both* (remote end of cable). This should result in activity on the received data (RD) line.

Solution - Rewire cable to satisfy hardware handshake. Even if you have configured the PC side for no hardware handshake, the remote device may need one or more handshake lines true for proper operation. In this case, it is often possible to *loop back* a true control output to the required control line inputs. On a DCE device, try connecting pin 6 (DSR) to pin 20 (DTR). On a DTE device, try connecting pin 20 (DTR) to pins 5 (CTS) and 6 (DSR).

Multiline Interrupt Driver

Programmer's Guide

Version 2.2

Rev 2.2b 8/22/88

MID FUNCTION REQUEST CALLS

	init port	xmit char	recv char	port status	buffer count	flush buffer	xmit block	recv block	system reset	set hs	set MCR	reset MCR	set/re diviso
AH	0	1	2	3	4	5	6	7	8	9	10	11	12
AL	init byte	char	•	•	1=tran 2=recv	1=tran 2=recv 3=both	•	•	•	hs cf byte	bits	bits	•
BX	•	•	•	•	•	•	buffer offset	buffer offset	•	•	•	•	•
CX	•	•	•	•	•	•	count	count	0	•	•	•	diviso
DX	port #	port #	port #	port #	port #	port #	port #	port #	•	port #	port #	port#	port#
ES	•	•	•	•	•	•	buffer segment	buffer segment	•	•	•	•	•

RETURN VALUES

	init port	xmit char	recv char	port status	buffer count	flush buffer	xmit block	recv block	system reset	set hs	set MCR	reset MCR	set/re diviso
AH	line status	line status	line status	line status	.	•	.	.	•	•	•	•	•
AL	modem status	•	char if avail.	modem status	.	•	.	.	•	•	MCR	MCR	•
AX	.	.	.	.	buffer count	•	return code	# chars remaining	•	•	•	•	•
CX	•	•	•	•	•	•	actual count	actual count	version#	•	•	•	diviso

INITBYTE SPECIFICATION

Bit Usage

7,6,5 Baud Rate(MIDHS)

000 = 110 (19.2k) baud
001 = 150 (38.4k) baud
010 = 300 (57.6k) baud
011 = 600 (115.2k)baud
100 = 1200 baud
101 = 2400 baud
110 = 4800 baud
111 = 9600 baud

4,3 Parity

X0 = no parity
01 = odd parity
11 = even parity

2 Number of Stop Bits

0 = 1 stop bit
1 = 2 stop bits

1, 0 Number of Data Bits

10 = 7 data bits
11 = 8 data bits

LINE STATUS REGISTER

Bit	Usage
7	Time out
6	Transmit shift reg. empty
5	Transmit hold reg. empty
4	Break detected
3	Framing error
2	Parity error
1	Overrun error
0	Data ready

MODEM STATUS REGISTER

Bit	Usage
7	Receive line signal detect
6	Ring indicator
5	Data set ready
4	Clear to send
3	Delta receive line signal detect
2	Trailing edge ring detect
1	Delta data set ready
0	Delta clear to send

LINE CONTROL REGISTER

Bit	Usage
7	Divisor latch access bit
6	Set break
5	Stuck parity
4	Even parity select
3	Parity enable
2	Number of stop bits
1,0	Word length select

MODEM CONTROL REGISTI

Bit	Usage
7,6,5	Not used
4	Loopback control
3	Out2 (IRQ gate)
2	Out1 (not used)
1	Request to Send (RTS)
0	Data Term. Ready (DTR)

MID Version 2.2

I. INTRODUCTION

Why MID?

Any programmer who has attempted to perform anything beyond the simplest communications task on an IBM PC (and most clones) can probably answer this question. Asynchronous communications has never been IBM's forte as attested to by the level of support offered in the standard BIOS rom. The standard BIOS is not interrupt driven which means that incoming data is easily lost if your program does not perform input in a tight, efficient loop. High speed multiport input is not feasible without an interrupt driver which catches each character as it arrives and buffers it for later retrieval by the application program. Even single port input (at any speed) is greatly simplified when an interrupt driver is used. Your program need not worry about losing data that may arrive while still processing a previous communication.

Although serial port output does not usually present the same problems of data loss as input does, interrupt driven output also enhances the operation of your application. Large or small blocks of data may be transferred to MID's buffer where transmission proceeds under interrupt control while your application is occupied elsewhere, perhaps monitoring input.

Design Philosophy

Several alternative methods of implementing device drivers were considered when MID was designed. The primary design criteria were:

- to provide fast, efficient multiport I/O

- to be as transparent as possible to existing software.

The installable device driver method was ruled out because the increased path length required for device calls degrades performance and because this method does not work with software using BIOS calls (which many programmers have resorted to). Also, MSDOS treats installable device drivers as either character devices or block devices. This complicates implementing a driver which supports both character and block calls.

MID is implemented as a BIOS extension/replacement which installs and remains resident when executed. This offers the following advantages:

- MID may be installed at any time, not just on reboot.

- MID is transparent to most existing software.

- Programmers familiar with the standard BIOS programming methods easily adapt to MID.

- A direct efficient path (software interrupt) is available for transfer of data between the application program and the device driver.

MID is optimized for concurrent multiport operation. When used with multiport boards, MID's design permits low overhead operation at high concurrent load levels at the expense of slightly increased overhead at low activity levels.

In addition to emulating the standard BIOS (int 14H) functions, MID provides extended functions which permit block I/O operations. These calls greatly enhance performance and should be used instead of the usual character calls whenever possible.

For efficiency, MID is written entirely in 8086/88 assembler language, hence, the native user interface to MID is the assembler language interface.

Assembler Language Interface

The primary program interface to MID is through a software interrupt. The exact interrupt used is user definable (See the User's Guide section on configuration). Services may be requested from assembler language by loading the registers with a function request number and parameters (if required), and performing a software interrupt, similar to requesting BIOS services. Section II describes the assembler language interface.

Higher Level Language Interface

The best way to perform serial I/O from higher level languages is to write a subroutine library which may then be linked with application programs. The subroutine library routines accept parameters from the calling program, set up the registers, and execute a software interrupt. Most popular compilers supply subroutines for setting the registers and performing software interrupts. Several sample subroutine libraries are included on the distribution disk which you may use as a model to implement functions for your favorite compiler. Section III describes the sample C language interfaces included with MID.

Sample Libraries

Appendix A lists the sample libraries currently supplied with MID. The sample libraries may be used as you please with the exception that the source code or unlinked object code may not be resold. However, you may incorporate the source code or the object code (by linking) into your application with no fees or royalties required. Sample libraries are included for your convenience and are not supported or guaranteed in any way.

II. MID FUNCTION REQUEST CALLS ASSEMBLER LANGUAGE INTERFACE

Serial input/output requests are made by loading registers with the appropriate parameters described below, and executing a software interrupt. The software interrupt number may be 14h or 60h through 67h and must correspond with the value given to the configuration program MCONFIG.

NOTE: The general convention is to number ports beginning with 1 in user documentation and beginning with 0 in programmer documentation. This document assumes port numbers begin with 0.

For all requests the DX register must be set to the port number to be accessed. This may be 0 through *n - 1* where *n* is the number of installed ports. The AH register must be set to a function request number. Functions 0 through 12 (0c hex) are currently valid. Calls with invalid function request numbers or out of range port numbers are ignored.

Functions 0 through 3 are essentially identical with standard BIOS int 14h functions. Functions 4 through 7 add additional block functions which greatly increase performance. Function 8 performs a reset, flushing all buffers and restoring the startup configuration. Function 9 allows program configuration of port handshake options and functions 10 and 11 permit manual control of handshake lines. Function 12 allows custom baud rates.

MID preserves all registers except AX and CX which may contain return values. The CX register is preserved by traditional calls (0 through 3).

The programming examples are code fragments for illustrative purposes and may assume unstated declarations or code. Programming examples are believed to be correct but have not been tested. Note that the examples don't do much in the way of error checking. The C examples call routines in the sample C interface library COMIO.C. Also refer to the sample interface libraries for programming examples.

Use of block functions when performing I/O at anything greater than keyboard speed will greatly enhance the performance of your application. Performance increases with increased block size.

MID's user functions are designed to be reentrant with respect to different ports. This permits operation in a multitasking environment. The exception is function 8 (System Reset), which performs a call to DOS which may cause reentrancy problems with DOS if called during another task's interrupted DOS call.

CHARACTER RECEIVED USER EXIT

A version of MID is supplied in Microsoft object module format that calls a user exit routine from the receiver interrupt routine every time a character is received. This can be useful for software protocols not supported by MID. See the sample assembler language source file on the disk (MIDUSER.ASM) for details.

STANDARD FUNCTIONS, CHARACTER I/O

FUNCTION 0 - Initialize Serial Port

DX = port number
AH = 0, initialize serial port
AL = initbyte

Returns:

AH = line status reg
AL = modem status reg

This function is used to configure the port and is similar to the standard BIOS call. Both the transmit and receive buffers are flushed. See the quick reference guide (center divider page) for a description of the init byte.

Assembler Example:

```
mov     ah,0        ; initialize function
mov     al,0E3h     ; 9600,n,8,1
mov     dx,3        ; do port 3
int     14h         ; call mid
```

C Example:

```
#define  NUMPORTS  8          /* number of ports installed */
#define  INITBYTE  0xE3      /* 9600,n,8,1 */

for (i=0; i < NUMPORTS; cominit (i++, INITBYTE));    /* do all eight ports */
```

FUNCTION 1 - Transmit a Character

DX = port number
AH = 1, transmit character request
AL = character to transmit

Returns:

AH = line status register
high bit (bit 7) set if transmit timeout

This function will transmit the character in the AL register and is similar to the standard BIOS call with the exception that the program will not wait for output unless MID's buffer is full. MID times out if buffer space does not become available within the timeout period specified to MCONFIG. The transmit buffer operates in a circular fashion and is transparent to the application.

Assembler Example:

```
again:
    mov     ah,1        ; transmit function
    mov     al,char     ; get character to send
    mov     dx,port     ; and port to send to
    int     14h         ; call MID
    test    ah,80h      ; check for timeout
    jnz     again       ; do it again
```

C Example:

```
    char string[] = "How now brown cow";

    for (i=0; string[i]; comout(port,string[i++])); /* send the whole string */

/* with error checking */

    if (comout(port,char))              /* transmit character */
        printf("device not ready\n");   /* transmit timeout */
```

FUNCTION 2 - Receive a Character

DX = port number
AH = 2, receive a character

Returns:

If no character available:
  AH = nonzero (high bit 1)
If a character is available:
  AH = line status register error bits (normally 0)
  AL = character

This function is similar to the standard BIOS function with the exception that it returns immediately to the calling program with the timeout bit set if no character is available. The error bits returned reflect the current contents of the line status register at the time of the call, and don't necessarily indicate an error in the returned character. MID's receive buffer operates in a circular fashion transparent to the application. If MID's circular buffer is full when an incoming character is available, MID will not read the character from the UART. This will cause the line status register overrun bit to be set upon receipt of the next character.

Assembler Example:

```
read:                  ; this will wait forever if no char comes in
    mov    ah,2        ; receive char function
    mov    dx,port        ; port to read
    int    14h            ; call MID
    test   ah,80h         ; was a character there?
    jnz    read           ; go try again
    mov    cbuf,al        ; save character
```

C Example:

```
int  port;        /* port number */
ch   char;        /* character buffer */

while ((ch = comin(port)) = = 0);/* wait for character */

/* same thing with timeout */

timer(0);                          /* start timer */
while ((ch = comin(port)) = = 0)
    if (timer(1) > TIMEOUT)            /* check timer */
        break;

/* same thing but polling */

for (port = 0; port < NUMPORTS; + +port)       /* poll all ports */
    if (ch = comin(port))                  /* if a character is there */
        buffer[port][index[port]+ +] = ch;/* stuff it in buffer */
```

FUNCTION 3 - Request Port Status

DX = port number
AH = 3, status

Returns:

AH = line status register
AL = modem status register

This function is similar to BIOS int 14h function 3. The status returned is the current status registers' contents.

Assembler Example:

```
mov     ah,3        ; status function
mov     dx,port     ; port number
int     14h         ; call mid
mov     stat,ax     ; and save status
```

C Example:

```
int status [NUMPORTS];      /* line stati */

for (i = 0; i < NUMPORTS; status[i++] = comstat(i));     /* getem all */
```

EXTENDED FUNCTIONS, BLOCK I/O

FUNCTION 4 - Return Buffer Count

DX = port number
AH = 4, return buffer count
AL = 1 to retrieve transmit buffer count
= 2 to retrieve receive buffer count

Returns:

AX = buffer count

If AL is set to 1, MID returns the number of characters currently in the transmit buffer awaiting transmission. If AL is set to 2, MID returns the number of characters in the receive buffer which have not yet been read by the application. This function is intended for use with block mode calls but may also be useful in character mode, in a polling routine, for instance. In a multiline environment, this approach is generally better than implementing wait style timeouts.

Assembler Example:

```
        mov     dx,NUMPORTS     ; set up countdown
loop:
        dec     dx
        js      exit    ; get out if done with all ports
        mov     ah,4    ; buffer count function
        mov     al,2    ; get receive buffer count
        int     14h     ; call mid
        test    ax,ax   ; any characters available?
        jz      loop    ; no, go check next port
        call    blkread ; yes, go read block
        jmp     loop
exit:                   ; go do something else for awhile
```

C Example:

```
/* same thing, get any waiting data */

char buffer [NUMPORTS][BUFLEN];

for (i = 0; i < NUMPORTS; i++)
{
    if (comcnt(i,2))
        cnt[i] += cominblk (i, MAX, buffer[i] + cnt[i], _showds());
}
```

FUNCTION 5 - Flush Buffer

DX = port number
AH = 5, flush buffer
AL = bit 0 set to flush transmit buffer
bit 1 set to flush receive buffer

Returns: Nothing

This function will discard any untransmitted characters remaining in the transmit buffer if AL bit 0 is set (AL = 1). If AL bit 1 is set (AL = 2), MID will discard any characters in the receive buffer that have not yet been read by the application. Both buffers may be flushed with one call by setting both bits (AL = 3). Note that function 0 also flushes both buffers.

Assembler Example:

```
; flush receive buffer in preparation for a block receive
    mov   ah,5      ; flush function code
    mov   al,2      ; receive buffer
    mov   dx,port
    int   14h       ; call mid
    mov   ah,1      ; transmit char function
    mov   al,ACK        ; send block request to remote device
    int   14h           ; port num is still in dx
```

C Example:

```
flush (port,2);         /* flush receive buffer */
comout (port,ACK);      /* request next block */

flush (port,3);         /* flush both xmit and recv */
```

FUNCTION 6 - Transmit Block

DX = port number
AH = 6, transmit block
ES = source buffer segment
BX = source buffer offset
CX = number of bytes to transmit

Returns:

AX = return code:
0 = complete
1 = partial transfer
CX = actual number of bytes transferred

This function will transmit the contents of the buffer (of length CX) pointed to by the ES BX register pair. The buffer contents are transferred to MID's local buffer where transmission proceeds under interrupt control.

The actual number of bytes transferred is returned in the CX register. MID will accept at most (transmit buffer size - 1) bytes assuming the transmit buffer is empty. If the transmit buffer is partially full MID may perform a partial transfer. The application program should generally check the return count and execute another block transmit call to transmit the remainder of the buffer if necessary.

Block transmit calls operate in a circular fashion and may be intermixed with character transmit calls.

Assembler Example:

```
mov    dx,port            ; port number
mov    ah,6               ; transmit block function
push   ds
pop    es                 ; buffer segment address
mov    bx,offset buffer   ; buffer offset
mov    cx,258             ; number of chars to send
int    14h                ; call mid
cmp    cx,258             ; check actual number sent
je     done               ; all sent
                          ; send rest of block
```

C Example:

```
char buffer [BUFLEN];  /* data to transmit */
int nout;              /* number of bytes transmitted so far */

nout = 0;              /* initialize */

while (nout < BUFLEN) /* until all characters have been accepted */
    nout + = coutblk (port, BUFLEN - nout, buffer, _showds());
```

FUNCTION 7 - Receive Block

DX = port number
AH = 7, block receive
ES = destination buffer segment
BX = destination buffer offset
CX = maximum number of bytes to receive

Returns:

AX = number of bytes remaining to be read
CX = actual number of bytes transferred

This function is used to request a block transfer from MID's receive buffer to the application program's buffer pointed to by the ES BX register pair. The lesser of CX or the actual number of bytes available will be transferred. If the input buffer is currently in a wrapped state two calls are required to read the entire buffer. The first call will read to the physical end of the buffer, and the next will return the remainder. Block reads operate in a circular fashion and may be intermixed with character read requests. When performing controlled block input such as an xmodem file transfer, it is good practice to flush the receive buffer (see Function 5) immediately before requesting the next block of data. This will avoid a wrapped buffer, as well as discarding any garbage accumulated from line noise.

Assembler Example:

```
mov   dx,port            ; port number
mov   ah,7               ; block receive function
push  ds
pop   es                 ; buffer segment address
mov   bx,offset buffer   ; buffer offset
mov   cx,BUFLEN          ; max to read
int   14h                ; call mid
mov   cnt,cx             ; save count
```

C Example:

```
cnt = cominblk(port,BUFLEN,buffer,_showds()); /* cnt = number read */
    /* note: cominblk() gets both pieces of wrapped buffer */
```

FUNCTION 8 - System Reset

AH = 8, reset function
CX = 0, if you wish to verify MID's presence

Returns:

CX = version of MID in binary, two decimal places implied

Function 8 causes MID to perform a complete reset to the state existing when initially installed. This may be necessary if software which reconfigures the interrupt structure has been run since MID was installed. All buffers are flushed, ports are reconfigured to their startup configurations, and the IRQ vector is reinstalled and enabled. It is good practice to call this function at the start of your program before calling any other functions. This function is performed if MID finds itself already installed when invoked from the system prompt.

This function returns the version of MID in binary with two decimal places implied. That is, version 2.55 would return as 00FF (hex), or 255 integer. If you set the CX register to zero prior to making this call, it is possible to determine with some certainty if MID is installed since standard BIOS calls generally preserve the CX register.

This function should recover from most ill-mannered software. This function performs a DOS call and should not be called in a multitasking environment when an interrupted DOS call might already be in progress.

Note: MID does NOT reread the configuration file MID.CF when reset. Values are reset to configuration as read from disk when MID was originally installed.

Assembler Example:

```
mov    ah,8    ; reset function
int    14h     ; call mid
```

C Example:

```
if (midreset() == 0)
{
    printf("Error - MID not installed");
    exit(8);
}
```

FUNCTION 9 - Set Handshake Mode

AH = 9, set handshake mode function
AL = handshake configuration byte
bit 0 = software transmit flow control enable
bit 1 = software receive flow control enable
bit 2 = hardware transmit flow control enable
bit 3 = hardware receive flow control enable

This function is used to enable or disable hardware or software flow control. In this version of MID software flow control means XON XOFF protocol and hardware flow control means DTR/DSR control. Bits 4 through 7 of the AL register are currently ignored but are reserved for future use and should be set to 0.

Assembler Example:

```
mov   dx,0    ; do port 0
mov   ah,9    ; handshake configuration function
mov   al,3    ; set full xon xoff protocol
int   14h     ; call mid
inc   dx      ; do port 1 (note: dx is always preserved)
mov   al,0    ; no handshake for this port
int   14h     ; note: some functions don't preserve ah register
```

C Example:

```
set_hs (port,0x0C);        /* set full hardware handshake */
```

FUNCTION 10 - Set UART MCR Bits

DX = port number
AH = 0A (hex), set modem control register bits
AL = modem control register bits to set to 1
bit 0 = Data Terminal Ready (DTR) control
bit 1 = Request to Send (RTS) control
bit 2 = Out 1 (UART Control line)
bit 3 = Out 2 (Resetting to 0 will disable the UART's interrupts)
bit 4 = Loopback control. See the 8250 data sheet for use
bits 5-7 Not used

Returns:

AL = new modem control register contents

This function is used to set selected bits in the UART's modem control register to 1. The contents of the AL register is or'ed with the UART's modem control register and the new contents are returned in the AL register. The purpose of functions 10 and 11 is to permit direct program control of the UART's hardware handshake signals. Unless MID is configured for no hardware handshake, MID will continue to raise and lower DTR as buffer conditions demand. DTR and RTS are also raised when a port is reconfigured (function 0), and when a system reset is performed (function 8).

Assembler Example:

```
mov   dx,0    ; do port 0
mov   ah,10   ; set MCR bits function
mov   al,1    ; turn DTR on
int   14h     ; call mid
```

C Example:

```
int bits = 2;
newmcr = mcr_set(port,bits);       /* raise RTS */
```

FUNCTION 11 - Reset UART MCR Bits

DX = port number
AH = 0B (hex), reset modem control register bits
AL = modem control register bits to reset to 0
    bit 0 = Data Terminal Ready (DTR) control
    bit 1 = Request to Send (RTS) control
    bit 2 = Out 1 (UART Control line)
    bit 3 = Out 2 (Resetting to 0 will disable the UART's interrupts)
    bit 4 = Loopback control. See the 8250 data sheet for use
    bits 5-7 Not used

Returns:

AL = new modem control register contents

This function is used to reset selected bits in the UART's modem control register to 0. The contents of the AL register is inverted and then and'ed with the UART's modem control register and the new contents are returned in the AL register.

Assembler Example:

```
mov   dx,0    ; do port 0
mov   ah,11   ; reset MCR bits function
mov   al,1    ; turn DTR off
int   14h     ; call mid
```

C Example:

```
int bits = 2;
newmcr = mcr_reset(port,bits);        /* lower RTS */
```

FUNCTION 12 - Read/Write baud rate divisor

DX = port number
AH = 0Ch Function number is 12.
CX = baud rate divisor, or 0 to read current divisor.

Returns:

CX = current baud rate divisor

This function permits configuration of the UART for nonstandard baud rates. Please refer to the sample program BRDTEST.C for examples of use. The actual resulting baud rate is also a function of the UART's clock input frequency (normally 1.8432 MHz). The formula is:

bit rate = clock / (16 * baud rate divisor)

This function leaves the data format alone so the normal procedure would be to first call function 0 (cominit()) to set the data format, then call this function (com_brd()) to set the baud rate.

Note: Not all libraries currently support this function.

C Example:

```
unsigned brd;

brd = com_brd(0,0); /* fetch current brd */
scr_rowcol(12,0);
printf("\rBaud rate divisor = %u",brd);

printf("\rEnter new baud rate divisor: ");
scanf("%d",&brd);
com_brd(0,brd);      /* set new brd */
```

III. C LANGUAGE LIBRARY

This section describes the routines contained in the sample C language subroutine library. Although I/O to two ports may work using traditional operating system calls or device type calls, the recommended way to perform serial I/O via MID is by calling routines such as those contained in COMIO.C. COMIO may be easily modified for any C compiler with a library function which provides a method of setting parameter registers and performing a software interrupt. Sample libraries for several popular C compilers are supplied with MID. See Appendix A for currently supplied libraries.

Further programming examples may be found in the section on the assembler language interface (Section II.) and in the test programs included on the disk.

Note that a successful return code from functions which perform output only means that the character(s) was successfully transferred to MID's buffer. If handshaking is enabled and the remote device is not accepting characters these functions will not return an error until MID's buffer is full.

Character Functions

comin()

```
int comin (port)        /* receive a char */
    int port;           /* returns char, or 0 if none */
```

```
example:    char c;
            c = comin(0);
```

comin() reads one character from the specified port. If no character is available, 0 is returned. Note that this function filters nulls and should not be used for reading binary data - use cominblk().

comout()

```
int comout (port,ch)  /* send a character out a port */
                      /* returns nonzero on error */
    int port;
    int ch;               /* character to send */
```

```
example:    comout(0,'A');
```

```
example:    if comout(1,'A')
                printf("device error");
```

comout() transmits one character out the specified port. comout() returns a zero on successful completion or nonzero if a transmit timeout occurs. This will normally occur only if handshaking is enabled and the remote device is not ready. A transmit timeout may also occur if the port is configured for a low data rate such that your program is sending characters to MID faster than the data rate permits transmission. Selecting the *short timeout* option when configuring the port with MCONFIG will normally prevent such timeouts.

rdcom()

```
int rdcom (port,buf,max) /* read all available chars from port to buf[] */
                         /* returns number of characters read */
    int port;            /* port number */
    char *buf;           /* buffer address */
    int  max;            /* maximum number of characters to read */
```

```
example:   char buffer[500];
           int cnt;
           cnt = rdcom(0,buffer,500);
```

rdcom() performs character reads to the indicated buffer as long as characters are available or until max characters have been read. It is considerably more efficient to use cominblk() for this purpose. This function calls comin() so the warning about nulls also applies here.

Block Functions

cominblk()

```
int cominblk (port,maxsiz,bufoff,bufseg)    /* receive a block, returns cnt */
    int port;
    unsigned maxsiz;     /* max characters to read */
    unsigned bufoff;     /* buffer offset */
    unsigned bufseg;     /* buffer segment address */
```

```
example:   char buffer[500];
           int nin;
           nin = cominblk(port,500,buffer,_showds());
```

cominblk() transfers at most maxsiz characters to the user buffer pointed to by bufoff and bufseg. cominblk() checks the return count and performs a second call if characters remain to be read automatically reading both parts of a wrapped buffer.

coutblk()

```
int coutblk (port,blksiz,bufoff,bufseg)     /* transmit block, returns cnt */
    int port;
    unsigned blksiz;     /* number of characters in buffer */
    unsigned bufoff;     /* buffer offset */
    unsigned bufseg;     /* buffer segment address */
```

```
example:   char buffer[500];
           int  nout;
           nout = coutblk(0,250,buffer,_showds());
```

The example would transmit the first 250 bytes of buffer[] out port 0. Note that if blksiz is greater than the free space in MID's transmit buffer, only buffer size characters will be transferred. Make another call to transmit the rest. The return count should ALWAYS be checked and any unaccepted characters resent. Repeated refusals may mean that the remote device has disabled transmission by sending an XOFF character (XON XOFF enabled) or by dropping DSR (hardware handshake enabled).

String Functions

prtcom()

```
int prtcom (port,string)  /* print string to serial port - as is */
                          /* returns nonzero on error */
    int port;             /* line number */
    char *string;         /* pointer to zero terminated string */
```

example: prtcom(0,"Hi there");

prtcom() is the same as prtcomr() except that no CRLF is appended. Zero is returned on successful completion, nonzero otherwise.

prtcomr()

```
int prtcomr (port,string) /* print string port, follow with crlf */
                              /* returns nonzero on error */
    int port;             /* line number */
    char *string;         /* pointer to zero terminated string */
```

example: prtcomr(0,"Hi there");

prtcomr() transmits a zero terminated string out the indicated port. A CRLF sequence is appended to the string. Zero is returned upon successful completion, nonzero on error.

General Functions

com_brd()

```
unsigned com_brd(port,brd);
    int port;
    unsigned brd;    /* new baud rate divisor, or 0 to read */
```

com_brd() permits configuration of the UART for nonstandard baud rates. Please refer to the sample program BRDTEST.C for examples of use. The actual resulting baud rate is also a function of the UART's clock input frequency (normally 1.8432 MHz). The formula is:

bit rate = clock / (16 * baud rate divisor)

This function leaves the data format alone so the normal procedure would be to first call function 0 (cominit()) to set the data format, then call this function (com_brd()) to set the baud rate.

Note: Not all libraries currently support this function.

comcnt()

```
int comcnt (port,which)        /* return buffer count */
    int port;
    int which;                 /* which buffer - 1=transmit, 2=receive */
```

example:
```
int c1;
c1 = comcnt(1,2);
```

comcnt() returns the number of characters in MID's buffer (transmit or receive, as specified by which) for the indicated port.

cominit()

```
void cominit (port,initbyte)   /* initialize a port */
    int port;
    int initbyte;              /* same as standard bios call */
```

example:
```
cominit(0,0xE3);
```

cominit() allows user configuration of serial ports. The example would configure port 0 for 9600 baud, no parity, eight data bits, and one stop bit. See the quick reference (center divider page) for a description of the initbyte.

comstat()

```
unsigned comstat (port)        /* return UART status registers */
    int port;
```

example:
```
unsigned stat;
stat = comstat(2);        /* get port 2 status */
```

comstat() returns the current UART status registers for the specified port. The line status register is returned in the high byte of the word returned, and the modem status register is returned in the low order byte.

flush()

```
void flush (port,which)        /* flush buffers */
    int port;
    int which;                 /* 1=transmit,2=receive,3=both */
```

example:
```
flush(0,3);        /* flush both buffers, port 0 */
```

A flush() call causes MID to discard any characters currently in the specified buffer(s) of the specified port.

mcr_set()

```
int mcr_set(port,bits) /* set mcr bits */
     int port;
     int bits;       /* see function 10 for bit descriptions */
```

example: mcr_set(0,0x02);

mcr_set() allows user control of UART handshake lines. The example would set the RTS bit thereby raising the UART's request to send line. This function returns the new contents of the modem control register.

mcr_reset()

```
int mcr_reset(port,bits)    /* reset mcr bits */
     int port;
     int bits;       /* see function 10 for bit descriptions */
```

example: mcr_reset(0,0x02);

mcr_reset() allows user control of UART handshake lines. The example would reset the RTS bit thereby dropping the UART's request to send line. This function returns the new contents of the modem control register.

midreset()

```
int midreset()              /* reset MID */
```

example: version = midreset();

midreset() results in a complete reset of MID restoring MID to startup conditions. midreset() returns the version of MID (two decimal places implied) or 0 if MID is not installed. It is good practice to call midreset() before performing any other functions.

set_hs()

```
void set_hs (port,hsconfig)     /* set handshake configuration */

    int port;               /* port number */
    int hsconfig;           /* handshake configuration byte
                  bit 0 = software transmit handshake enable
                  bit 1 = software receive handshake enable
                  bit 2 = hardware transmit handshake enable
                  bit 3 = hardware receive handshake enable      */
```

example: set_hs(0,3); /* enable full software handshake for port 0 */

set_hs() permits program configuration of a port's handshake mode. Setting bit 0 through 3 enables the corresponding flow control. A zero bit value disables the corresponding flow control.

IV. BASIC INTERFACE

The file BASCOMIO.O may be linked with a Microsoft Compiled Basic or Microsoft Quick Basic program. This provides a function callable from basic. This interface cannot be used with Microsoft interpreted basic. There is no equivalent of the basic *on COM condition*. Your program must periodically poll the port(s) in question.

Following is a description of the calling sequence. Note the possibility of two return values (ctldata% and rtncode%). Buffer$ only provides an address; MIDSIO does not inspect nor alter the string control information. In the case of block calls, the return count in ctldata% should be examined to determine valid string length. It may be helpful to also reference the MID function descriptions in the assembler documentation.

Functions 10, 11, and 12 are not currently supported by the basic interface.

CALL MIDSIO(buffer$,port%,function%,ctldata%,rtncode%)

where:

buffer$ = character space for input/output

port% = port number (0-7)

function% = function code	ctldata% = function parameter is dependent upon function
0 = initialize serial port	0 - call: baud/fmt (E3=>9600,n,8,1) rtn: lsr/msr (line/modem status regs)
1 = transmit a character	1 - rtn: lsr/0
2 = receive a character	2 - rtn: lsr/0
3 = request port status	3 - rtn: lsr/msr
4 = request buffer count	4 - call: 1=>xmit bufr, 2=>recv bufr rtn: buffer character count
5 = flush buffer(s)	5 - call: 1=>xmit bufr,2=>recv bufr,3=>both
6 = transmit a block	6 - call: number of chars to xmit rtn: number chars transferred
7 = receive a block (detects & reads past buffer wrap)	7 - call: max number of chars to recv rtn: number chars transferred
8 = system reset	8 - not used
9 = set handshake mode	9 - call: handshake config byte

rtncode% = return code information returned from MID. Interpretation depends upon function called:

- 0 = normal return
- 1 = invalid function number
- 2 = invalid port number
- 3 = no char received from function 2
  = no char transmitted fun. 1 (timeout)
- 4 = invalid buffer number for function 4 or 5
- 5 = incomplete transfer from function 6
- n = no. chars remaining in MID's bufr from function 7
- n = version from function 8 reset, or 0 if not installed

V. TESTING AND PERFORMANCE

Two simple test programs are included on the disk.

Test Programs

The test programs are designed to be used with terminals, or PCs emulating terminals attached to the serial ports. The terminals should be set for line wraparound for programs TEST1 and BIGTEST. Output may also be monitored on a breakout box.

The test programs are compiled for user service interrupt 14h. You must configure MID accordingly, or recompile the test programs.

Make sure that MID installed. Invoke the test by typing the program name followed optionally by one or two positional parameters. *Unless overridden all test programs default to one port operation and leave the ports' current configuration unchanged.* These defaults may be overridden by optional parameters:

First parameter = number of lines active (*defaults to 1*).

Second parameter = serial port configuration byte, in hex.
This is the same as IBM, eg. 83 means 1200,n,8,1
If you wish to specify this parameter, you also must include
the first parameter.

For example, "BIGTEST 6 83" will invoke the full screen write test with six lines active at 1200 bps.

Test Program Operation

Both programs stream data out all active ports and monitor input from any one chosen port. Common characteristics are:

TO QUIT: press the escape key

TO MONITOR INPUT: press one of the number keys to monitor the corresponding line on the pc's screen. Line numbers start at 0.

Operation at the terminals differs with each program as follows:

TEST1.EXE - This program continuously performs a block write of an 80 character test pattern out each port. Receipt of a lowercase 'l' (for left) from the same port will shorten the pattern to 79 characters and a lowercase 'r' will lengthen it to 81. Type a lowercase 'm' at the terminal to restore it to 80. TEST1 monitors input with character input calls.

BIGTEST.EXE - This program continuously performs a block write of a full 1920 byte screen buffer to each port. Block reads are used for monitored line input. This program achieves the highest throughput of all the test programs. BIGTEST works with any buffer size configuration and illustrates the performance advantage of large buffers.

Loopback Testing

Loopback testing is one way to observe the effects of interrupt overhead on the foreground program, in this case the test program. Output from any of the test programs may be looped back by jumpering from the transmit pin of one port to the receive pin of another (or the same) port. BIGTEST.EXE is the most suitable for this since it uses block calls for input and is considerably faster on input than the other test program which uses character input functions. Even so, at some combination of ports active and baud rate, the monitored line receive buffer will overflow. Overflow occurs because the test program can't write the characters to the PC's screen as fast as they come in. Enabling xon xoff protocol or hardware handshaking (use MCONFIG, and reboot) will prevent overruns. Note that this only works with a line looped back to itself, or with two lines transmitting to each other.

Overhead and Capacity

NOTE: This discussion does not constitute a performance claim or guarantee. Parasoft does its best to provide high performance software, but given the variety of hardware and software environments where MID might be used, Parasoft makes no claims as to performance or suitability for any particular purpose.

The following discussion is based on observations using eight ports on one board/IRQ. Overhead and capacity are significantly impacted when configured for multiple IRQs. In general, multiple IRQs will reduce overhead at low activity levels and increase overhead at high activity levels. Peak capacity will be significantly reduced for n ports configured to multiple IRQs as compared to n ports on one IRQ.

When running the test programs with output looped back, you may notice that the monitored input may surge somewhat under heavy load, even stopping under extreme loads. This is because of the varying interrupt overhead as block transmits out the ports overlap. The test programs place a very high I/O load on the pc, much higher than typically encountered in actual use.

On a standard PC (Norton 1), MID approaches 100% interrupt overhead at a total load of (very) roughly 80,000 bits per second, or 8 ports, half duplex, at 9600 baud. Because of MID's design, this occurs at approximately half of system capacity. In other words, overhead per character decreases as total load increases.

MID's *theoretical single IRQ* peak receive bandwidth (Version 1.20, 4.77 mhz 8088) is over 150,000 bps. When reduced 25% to allow for instruction fetch waits, timer interrupt overhead, etc., we are still left with 112,500 bps or 12 ports receiving simultaneously at 9600 baud. MID's transmit bandwidth has not been calculated, but probably exceeds the receive bandwidth. Of course, a receive load would reduce available transmit bandwidth. These figures all assume no handshaking - enabling handshake reduces available bandwidth slightly.

BIGTEST.EXE, when used with large buffers, sustains an average aggregate transmission rate of about 58,000 bps on a Norton 1 machine, eight ports on one board/IRQ, and no input occurring. This is 75% of theoretical maximum, and the other 25% can be attributed to test program overhead including repeated transfer of the pattern buffer contents to MID's buffers for transmission. Enabling XON XOFF without invoking it causes only a slight degradation in performance. BIGTEST regularly flushes non-monitored ports to prevent stoppage of looped back ports.

MID and Other Interrupts

MID can coexist with other hardware interrupt routines provided the following:

> HIGHER PRIORITY IRQS: total service time (or until EOI is issued) must be less than character time of highest baud rate port and interrupt frequency must permit MID time to service input from ports. MID's bandwidth will be reduced in proportion to the cpu requirements of the higher priority IRQ.
>
> LOWER PRIORITY IRQS: maximum interrupt disable time must be less than character time of highest baud rate port and interrupt must tolerate service delay due to MID's interrupt overhead.

There have been no reported problems with standard PC hardware interrupts. Software which intercepts IRQ 0 (PCDOS interrupt 8, timer) or PCDOS interrupt 1C (user timer control, software interrupt) may cause loss of data if interrupt service time is excessive.

Warning: If you use a timer interrupt to drive your program, don't wait too long before issuing an EOI to the 8259 interrupt controller. The timer is the highest priority interrupt and effectively masks all other interrupts even though you enable interrupts with an STI instruction.

Interrupt Disable Times

Note: the following figures were calculated based on MID version 1 and are not adjusted for possible prefetch inefficiencies. Disable times for MID version 2 have not been calculated but should be comparable. All figures are based on a 4.77 mhz 8088 cpu.

MID's IRQ interrupt routine reenables interrupts immediately upon entry and leaves them enabled until exiting, at which time interrupts are disabled for approximately 70 clocks (15 microseconds) during the end of interrupt sequence. Under heavy loads, interrupts are enabled 100% of the time.

MID's User Service interrupt routine reenables interrupts immediately upon entry and disables interrupts only when necessary. The longest period with interrupts disabled occurs during the transmit character function (function 1) and is approximately 150 clocks or 32 microseconds. The block transmit function (function 6) disables interrupts for approximately 25 microseconds. All other disable periods are believed to be shorter.

VI. MULTILINE PROGRAMMING TECHNIQUES

Concurrent multiport operation is simply impossible without an interrupt driver such as MID. MID provides the nearly instantaneous service that the ports require and frees the application program to process data at a higher block or message level. However, effective multiport programming requires a multitasking environment, or at least a multitasking approach to application design.

Simple Round Robin

If the communications tasks are not time sensitive and high performance is not a major design criterion, a simple round robin scheme may be adequate. The application program's main control routine simply loops on the port number and services each port in order according to the needs of the port. State tables may be implemented to remember the status of each line between services. The test programs included in source form on the distribution disk illustrate this approach. The major shortcoming of this approach is difficulty in servicing ports in a time sensitive and regular fashion.

Simple Multitasking

A more sophisticated version of the round robin scheme uses the PC's timer interrupt to invoke protocol drivers for each line. This is, in fact, a limited form of multitasking and provides an additional level of automatic control between MID and the application program. This permits lengthy procedures such as error correcting file transfers to proceed concurrently on multiple lines largely unattended by the application program. Driving with the timer interrupt provides better performance and eases implementation of time sensitive functions such as timeouts. The PC's 8259 interrupt controller must be reprogrammed to alter interrupt priorities or else the timer interrupt will preempt the serial port interrupts resulting in lost data.

Operating Environments

Products such as **DesqView** (Quarterdeck Office Systems) provide multitasking capability. A window(s) may be set up to process communications in the background. A developer's toolkit is available which even provides intertask communications, task initiation, etc. MID works with DesqView and may be installed before DV (any window may access ports), or may be installed within a window. If installed within a window, MID disappears when the window is closed. Two windows cannot share the same IRQ. The last installed MID copy will preempt a copy of MID in another window with the same IRQ.

MID does *not* currently work with Microsoft Windows 386. MID installs fine, and operates for a time (about three minutes). Then Windows retakes control of the hardware interrupt line because it detects no activity on the serial ports through the standard BIOS, which MID has replaced. This is in spite of the fact that MID obtains the interrupt vector through a DOS level call, and Windows *should know* that MID requires the vector until released. In short, Windows is trying to be smart, but doing it in a very dumb way. It assumes that all serial devices are sharable resources. I will refrain from further comment.

A more complete discussion of data communications protocols and multiline programming techniques is available in the excellent book *Data Communications Software Design* written by *Malcolm G. Lane* and published by *Boyd and Fraser (1985)*.

APPENDIX A - SAMPLE LIBRARIES

Sample program interface libraries are supplied for several popular language development systems. Parasoft currently uses C-WARE's DeSmet C development system (version 2.4) for internal C language development, and this library is the only library actually tested by Parasoft. The remaining libraries have been graciously supplied by MID users but have not been tested by Parasoft and may not reflect the latest changes or additions. In some cases, additions have been made but not tested.

Where object modules are included, the code has generally been generated for the small memory modem (1 data/stack segment and 1 code segment). If you are using a different memory model you will need to recompile the source file for the appropriate memory model.

Tested sample libraries:

C-WARE DeSmet C
Source code is COMIO.C
Object module is COMIO.O

Untested sample libraries:

Microsoft C
Source code is MSCOMIO.C
Object module is MSCOMIO.OBJ
Alternate library MSLCOMIO.C
Programming example is MBIGTEST.C

Lattice C
Use Microsoft library MSCOMIO.C

Mark Williams C
Source code is MWCOMIO.C

Borland's Turbo Pascal
Source code is COMIO.PAS
This library differs somewhat from the C libraries.

Microsoft (and IBM) Compiled Basic, Microsoft Quick Basic
Source code is BASCOMIO.ASM
Object module is BASCOMIO.OBJ (must be linked with runtime)

APPENDIX B - COMMON PROBLEMS AND SOLUTIONS

Problem: Computer dies when MID is installed.

Solution: Check that port addresses given to MCONFIG correspond with hardware settings. If your board has less than eight ports, make sure that you have configured MID for the correct number of ports.

Problem: MID installs ok, but nothing happens when test programs are run.

Solution: Check that IRQ given to MCONFIG corresponds with hardware setting. Make sure that no other hardware is configured for the same IRQ. Make sure you have configured MID for user service int 14h. Reboot and try again. Cycle power and try again.

Problem: MID operates ok on freshly booted PC, but later on output drops characters and input doesn't work at all.

Solution: Ill mannered software has masked MID's irq or replaced the irq vector. Perform a reset function, either from keyboard or programmatically. ,

---

Problem: Remote device will not transmit, xon xoff enabled.

Solution: Remote may be waiting for xon. Transmit an xon character.

Problem: MID stopped transmission, xon xoff enabled.

Solution: Remote sent xoff and never sent xon to restart transmission, or xon got lost or garbled. Check ready status of remote device and send xon (control-Q) from remote.

Problem: MID won't transmit, you are using device calls, redirection, or the CTTY command.

Solution: The operating system may require that the DSR and/or CTS hardware handshake lines be true before transmitting data. Check these lines with your breakout box, or loop them back from a true output signal (DTR, RTS, eg.).

ASCII CHART

Dec	Oct	Hex	Char	Ctl	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char
00	000	00	NUL	^@	032	040	20	SP	064	100	40	@	096	140	60	'
01	001	01	SOH	^A	033	041	21	!	065	101	41	A	097	141	61	a
02	002	02	STX	^B	034	042	22	=	066	102	42	B	098	142	62	b
03	003	03	ETX	^C	035	043	23	#	067	103	43	C	099	143	63	c
04	004	04	EOT	^D	036	044	24	$	068	104	44	D	100	144	64	d
05	005	05	ENQ	^E	037	045	25	%	069	105	45	E	101	145	65	e
06	006	06	ACK	^F	038	046	26	&	070	106	46	F	102	146	66	f
07	007	07	BEL	^G	039	047	27	'	071	107	47	G	103	147	67	g
08	010	08	BS	^H	040	050	28	(	072	110	48	H	104	150	68	h
09	011	09	HT	^I	041	051	29	)	073	111	49	I	105	151	69	i
10	012	0A	LF	^J	042	052	2A	•	074	112	4A	J	106	152	6A	j
11	013	0B	VT	^K	043	053	2B	+	075	113	4B	K	107	153	6B	k
012	014	0C	FF	^L	044	054	2C	,	076	114	4C	L	108	154	6C	l
013	015	0D	CR	^M	045	055	2D	-	077	115	4D	M	109	155	6D	m
014	016	0E	SO	^N	046	056	2E	.	078	116	4E	N	110	156	6E	n
15	017	0F	SI	^O	047	057	2F	/	079	117	4F	O	111	157	6F	o
016	020	10	DLE	^P	048	060	30	0	080	120	50	P	112	160	70	p
017	021	11	XON	^Q	049	061	31	1	081	121	51	Q	113	161	71	q
018	022	12	DC2	^R	050	062	32	2	082	122	52	R	114	162	72	r
019	023	13	XOFF	^S	051	063	33	3	083	123	53	S	115	163	73	s
020	024	14	DC4	^T	052	064	34	4	084	124	54	T	116	164	74	t
021	025	15	NAK	^U	053	065	35	5	085	125	55	U	117	165	75	u
022	026	16	SYN	^V	054	066	36	6	086	126	56	V	118	166	76	v
023	027	17	ETB	^W	055	067	37	7	087	127	57	W	119	167	77	w
024	030	18	CAN	^X	056	070	38	8	088	130	58	X	120	170	78	x
025	031	19	EM	^Y	057	071	39	9	089	131	59	Y	121	171	79	y
026	032	1A	SUB	^Z	058	072	3A	:	090	132	5A	Z	122	172	7A	z
027	033	1B	ESC	^[	059	073	3B	;	091	133	5B	[	123	173	7B	{
028	034	1C	FS	^\	060	074	3C	<	092	134	5C	\	124	174	7C	|
029	035	1D	GS	^]	061	075	3D	=	093	135	5D	]	125	175	7D	}
030	036	1E	RS	^^	062	076	3E	>	094	136	5E	^	126	176	7E	~
031	037	1F	US	^_	063	077	3F	?	095	137	5F	_	127	177	7F	DEL

BUG REPORT

While we have tried to assure this manual is error free, it is a fact of life that works of man have errors. We would request of you to detail any errors you find on this Bug Report and return it to us. We will correct the problems and send you new manual pages. This applies to both manual and software errors. Return to:

Industrial Computer Source
Attn: Manuals Department
Box 23058
San Diego, CA 92123
(619) 279-0084

Name ____________________________________

Company__________________________________

Address _________________________________

Address__________________________________

City________________________ State__________ Zip__________

Phone ___________________________________

What product? ___________________________

What manual revision?____________________

Please list the pages and errors below and mail to the above address. Thanks.

____

COMPUTER HARDWARE CONFIGURATION

1. PC MOTHERBOARD (E700)

Switch Position

1 OFF
2 OFF
3 OFF
4 ON

2. FAST COM4 (E700-3)

Base Address	IRQ Level
Switch Position	Switch Position
1 ON	1 OFF
2 ON	2 OFF
3 OFF	3 OFF
4 ON	4 OFF
5 OFF	5 ON
6 ON	6 OFF
7 ON	7 OFF
8 ON	8 ON

3. GPIB II (E700-4)

Switch Position

1 ON
2 OFF
3 ON
4 OFF
5 ON
6 ON
7 ON

4. OPTO 22 (E700-5)

Jumpers

A - 3 SHORT
A - 4 SHORT
A - 5 OPEN
A - 6 SHORT
A - 7 SHORT
A - 8 SHORT
A - 9 OPEN
A - 10 OPEN